ENG450 – Engineering Internship

Final Report

Final Report

FMG - Pilot Plant Upgrade Project

and

Curtin University - Green Electrical Energy Project

18th November 2011

Jeremy Karasavas

Murdoch University

Division of Science and Engineering

School of Engineering and Energy

South Street Campus

South Street

Murdoch

Western Australia, 6150

+61 (08) 9360 6000

Executive Summary

ICON Technologies is a software development company that specialises in measurement, automation, and control. ICON designs custom built systems for its clients using the LabVIEW platform, and combines modern hardware with professionally designed custom software to provide systems that exceed clients' expectations.

Whilst undertaking an internship with ICON technologies, two primary projects were worked upon. The first project was for Fortescue Metals Group Ltd (FMG), and involved developing a LabVIEW based control system for a fines-classification pilot plant. This plant is primarily used for research and development purposes, and is designed to process batch samples of collected materials so that an analysis can be conducted upon the performance of both current and envisaged control strategies. The HMI for the pilot plant has been developed by ICON Technologies, and has subsequently been approved by the client. The HMI is fully functional, with the underlying code to control user interaction being appropriately implemented. Although progress on the project was initially going smoothly, the project was required to be put on hold owing to budget constraints on the client's end.

The second project undertaken as part of the internship placement was for Curtin University, and involved the design and development of a LabVIEW-based monitoring system for Curtin's Green Electrical Energy Plant. This plant is primarily used for research purposes, and is utilised in laboratories and workshop classes for relevant engineering courses. At the time of Curtin University contracting ICON Technologies to work on the system, no monitoring system was in place. Curtin is aiming to have a computer-based monitoring system up and running by the first semester of 2012 so that the laboratory can be incorporated into the course-work for relevant engineering classes.

At the present time, a large amount of the Curtin Green Energy Project has been completed. A test setup has been implemented for the purpose of developing the system, and the system is currently in operation within the testing environment. Functional server and client programs have been developed for the system, and additional features are in the process of being implemented and refined to the client's specifications.

Upon completion of the Murdoch University Internship Program, both the Pilot Plant Upgrade Project, and the Curtin Green Energy Project are in their software development stages. The Pilot Plant Upgrade project has been put on hold, but is in a state where work can resume following approval from the client. The Curtin Green Energy Project is making steady progress, and is expected to be completed in the following few weeks. Throughout the internship program, numerous opportunities have been provided to apply the skills and professionalism taught at university to real-world engineering applications. The technical knowledge and industry skills that have been obtained through working with ICON Technologies will undoubtedly prove to be invaluable asset throughout future work within the engineering industry.

Table of Contents

Table of Figures

Table of Tables

1 Introduction

ICON Technologies is a software development company that specialises in measurement, automation, and control. ICON Technologies develops systems using the LabVIEW platform and implements these systems through the use of specialised hardware, known as Programmable Automation Controllers (PACs). Programmable Automation Controllers are compact controllers that combine the features and capabilities of a PC-based control system with that of a typical Programmable Logic Controller (PLC) [7]. ICON Technologies designs state-of-the-art, custom built systems for its clients by integrating the latest commercial off-the-shelf hardware with professionally engineered custom software. [6]

Whilst undertaking an internship with ICON technologies, two primary projects were worked upon. The first project was for Fortescue Metals Group Ltd (FMG), and involved developing a LabVIEW based control system for a material-processing pilot plant. The second project was for Curtin University, and involved the design and development of a LabVIEW-based monitoring system for Curtin's Green Electrical Energy Plant (GEEP). The following report aims to document the progress made on each of these projects and provide an indication of the work performed as part of the internship placement.

2 FMG – Pilot Plant Upgrade Project

2.1 Project Overview

2.1.1 Project Scope

Fortescue Metals Group Ltd (FMG) owns a fines-classification pilot plant that is used primarily for research and development purposes. The plant is designed to process batch samples of collected materials so that an analysis can be conducted upon the performance of both current and envisaged control strategies. Control strategies that are found to be efficient can then be applied to higher level processing applications in order to increase performance and product quality.

The pilot plant has been in operation for several years and has been configured to operate under a LabVIEW-based control environment. Compact Fieldpoint modules are currently used to handle the system's inputs and outputs, and both the control system and user-interface have been implemented through LabVIEW. FMG has recently discovered that there are a number of issues with the current software, and has subsequently determined that it is appropriate for this software to be rewritten. Additionally, FMG has decided to relocate the pilot plant from Cloudbreak to Karratha. A variety of new hardware components are also being installed so that the functionality of the plant can be improved.

ICON Technologies has been contracted by FMG to provide the required hardware for the plant, as well as to construct the control system environment in LabVIEW. The software prepared as part of this project is required to act as a stand-alone control system, facilitating: I/O manipulation, control loop implementation, user interfaces on multiple devices, system monitoring, alarm monitoring, and data logging capabilities. The software is required to be compatible with Microsoft Windows based client computers, as well as with FMG's Windows-7 enabled remote tablets.

Since ICON Technologies will primarily be concerned with the software design and hardware implementation aspects of the project, appropriate communication needs to be maintained with several other companies that are responsible for concurrent parts of the project. FMG has assigned 'Proteus EPCM Engineers' the job of designing the modified plant, as well as specifying the overall requirements of the control and monitoring system. A secondary company, RCR Tomlinson, has been contracted by FMG to organise and facilitate the physical relocation of the plant.

2.1.2 Required Tasks

ICON Technologies' fundamental role in FMG's pilot plant upgrade project is to design a LabVIEW-based control system for the 'to-be-upgraded' pilot plant, as well as to supply appropriate LabVIEW-compatible hardware that can facilitate computer-based interaction with the system. After numerous meetings with Craig Bartle (a representative from FMG), the functionality that FMG is expecting from the proposed control system has been clearly expressed. However, it has also been expressed that internal constraints relating to budget and project-approval will limit the work that can immediately be performed by ICON Technologies towards developing the control system. ICON Technologies has agreed to begin work on the project with knowledge of these constraints, and understands that many of the tasks required to complete the project to FMG's specifications are not yet required to be completed.

At the completion of the project, FMG is expecting the pilot plant's control system to support all of the features and capabilities that are listed below. (Detailed explanations of the features and capabilities that have been carried out by ICON Technologies are presented in section 2.2.)

1) A functional user-interface for the control-system that allows the system to be efficiently monitored and controlled. (Human Machine Interface [HMI])
2) The ability to acquire data from each of the system's inputs, and pass them to the control-system for processing and monitoring. (Data Acquisition)
3) The ability to control outputs from the system based upon user-defined settings, as well as measurements collected from the system through digital and analogue input channels. (Plant Control)
4) The ability to log data relating to the plant's operation. This includes all inputs to the control system, as well as outputs from the control system. (Variable Logging)
5) The ability to recall and review collected data so that it can be post processed and analysed. (Historical Trending Screen)
6) The ability for the control system to automatically generate alarms based upon pre-defined characteristics of system parameters. Appropriate methods for notifying the user of active alarms are also required to be implemented. (Alarm Generation and Monitoring)

The features listed above outline the key elements that FMG requires in the pilot plant's control system; however, not all of these tasks have been assigned to ICON Technologies. Although it is expected that ICON Technologies will carry out these tasks in the future, only a subset of these tasks have currently been assigned to ICON Technologies for completion. (Details relating to the current completion status of the project are presented in section 2.1.4.)

At this point in time, ICON Technologies has been instructed by FMG to provide a hardware specification for the system, completely build the system's user-interface in LabVIEW (including user interaction code), and construct a functional historical trending screen. For the purpose of this report, these will be considered the 'required tasks' for the project, as the assignment of any future tasks is completely at the discretion of FMG.

2.1.3 Initially Predicted Timescale of Events

After discussing the project with representatives from FMG, they expressed that the pilot-plant would ideally be in the commissioning stage by December 2011. However, it was clearly stated that there was uncertainty associated with the project going forward, and that the work performed would be dependent upon the ability of FMG to receive the required project budget from upper management.

Owing to the uncertainty of the project's timeline, building a Gantt chart to represent proposed completion dates for key project-milestones was of minimal practical use. Specifically, budget-related influences to project timelines were completely out of ICON Technologies' control and could not be known ahead of time. However, since it was a requirement of the internship that a Gantt chart be presented within the project plan, a Gantt chart was constructed under the desirable, yet unrealistic, assumptions that pre-warned budget-related delays would not occur, and that ICON Technologies would be later contracted by FMG to perform the full scope of the work. This timeline of events was constructed purely to comply with a requirement of the internship, and it was known prior to its construction that it would be of limited value to the project. Subsequently, this Gantt chart was not referenced to throughout the design process. The aforementioned Gantt chart can be seen in Figure 1 below.

	Task Name	Duration	Start	Finish
1	**Preparatory Work**	10 days	Mon 8/08/11	Fri 19/08/11
2	**Hardware Specification**	10.5 days	Mon 8/08/11	Mon 22/08/11
3	**Preliminary Client Consultations**	10 days	Wed 10/08/11	Tue 23/08/11
4	**Software Specification**	10.5 days	Mon 22/08/11	Mon 5/09/11
5	**Software Design**	10.5 days	Mon 22/08/11	Mon 5/09/11
6	**Software Development**	47 days	Tue 30/08/11	Wed 2/11/11
7	HMI	12.5 days	Tue 30/08/11	Thu 15/09/11
8	Popup Window Development	21 days	Wed 7/09/11	Wed 5/10/11
9	User Interaction Code	41 days	Wed 7/09/11	Wed 2/11/11
10	Control Loop Implementation	23 days	Mon 3/10/11	Wed 2/11/11
11	Trending Screen	19 days	Mon 19/09/11	Thu 13/10/11
12	Variable Logging	23 days	Mon 3/10/11	Wed 2/11/11
13	Alarm Monitoring	17 days	Tue 11/10/11	Wed 2/11/11
14	**Unit Testing**	31 days	Wed 21/09/11	Wed 2/11/11
15	**System Testing**	4 days	Fri 4/11/11	Wed 9/11/11
16	**Factory Acceptance Testing**	7 days	Fri 11/11/11	Mon 21/11/11
17	**Commissioning**	12 days	Mon 5/12/11	Tue 20/12/11
18				
19	Technical Investigations into Relevant Technologies	57 days	Tue 16/08/11	Wed 2/11/11

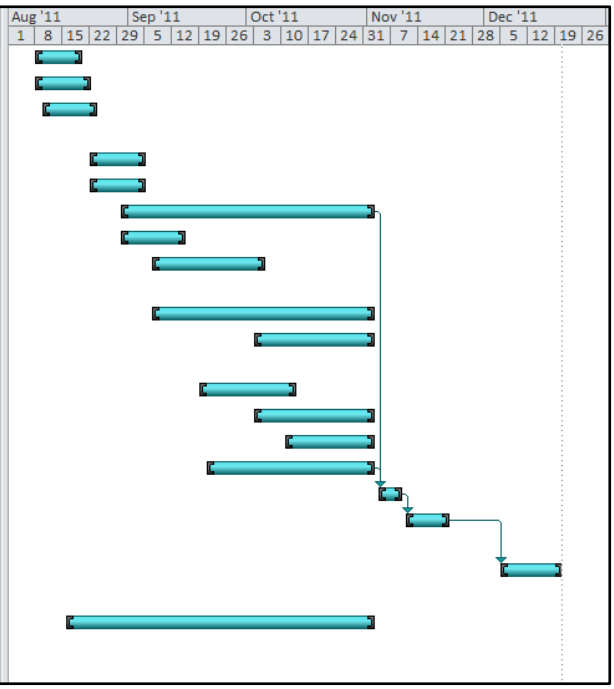

Figure 1 - FMG Project: Gantt Chart

2.1.4 Current Completion Status

At the time of this report's publication, only a small subset of the total software development tasks have been completed. In the early stages of the project, FMG indicated that they were yet to secure an appropriate budget for the project. Subsequently, it was expressed that work on the project may have to be delayed if the necessary funding did not get approved. ICON Technologies received an initial purchase-order from FMG, which has since been fully debited with chargeable work. Following the completion of this purchase order, FMG contacted ICON Technologies to notify them that issues had arisen with securing a budget for the project. Although it was expressed that ICON Technologies would likely be responsible for completing the remaining software development components in the future, a follow-up purchase order has not yet been issued, and the project has subsequently been put on hold.

During a preliminary meeting with representatives from FMG, discussions took place relating to the work that would be performed under the initial purchase order. Owing to the uncertainty associated with the project's budget, FMG expressed that the purchase order should be spent on work that can have its benefits physically seen so that the work performed could form part of a proposal to upper management to extend the project's budget. From this standpoint, it was decided that spending time on underlying code relating to data acquisition, data logging, controller implementation, and alarm development was not desirable since the benefits of the work would not be evident until a substantial portion of the development stage had been completed. After collaborating with FMG, it was determined that the best way to spend the initial purchase order was to develop a functional Human Machine Interface (HMI) for the plant in LabVIEW.

At the current point in time, the HMI for the pilot plant has been completed. The HMI is fully functional, with the underlying code to control user interaction being appropriately implemented. Since there is no legitimate data going through the HMI, simulated inputs and outputs have been implemented so that its functionality can be immediately demonstrated. The HMI consists of three primary screens, each of which has been reviewed and approved by FMG.

As it stands, the project is currently in a position where work can immediately recommence following the creation of a secondary purchase order. However, until FMG is in a position to financially commit to the project, no further work is able to be performed.

2.2 Current Progress

2.2.1 Hardware Specification

Since many additional hardware components are planned for installation within the pilot-plant, the Compact Fieldpoint modules that are currently in operation will not be sufficient to support the additional I/O. As such, it is necessary for additional LabVIEW-compatible hardware to be purchased so that the plant can be interfaced with through the LabVIEW-based control system. Although FMG chose to allocate the job of supplying a hardware specification to Proteus, ICON Technologies was in a position to recommend a hardware configuration to Proteus since the selected hardware correlates closely with the capabilities of the control system.

In order for a hardware specification to be developed, Proteus provided ICON Technologies with the relevant P&ID's and electrical wiring diagrams for the proposed upgraded pilot plant. FMG expressed that these diagrams were not finalised, and that it was expected that a large amount of digital I/O would be removed in future revisions.

To calculate the total amount of I/O required for the plant, each P&ID and electrical wiring diagram was reviewed, and a list was developed that outlined each of the plant's available inputs and outputs. The total I/O count that was inferred from the supplied P&ID's and electrical wiring diagrams can be seen in Table 1 below.

TOTAL I/O		
	P&ID	**Wiring Diagram**
DI:	63	**104**
DO:	21	**68**
AI:	33	**35**
AO:	17	**22**

Table 1 - Total I/O Count

As can be seen from Table 1, the analysis of the wiring diagrams returned a substantially higher I/O count than was inferred from the P&ID's. After presenting this observation to FMG, they expressed the view that the wiring diagrams showed redundancies that could not be seen on the P&ID's, and subsequently formed a more accurate representation of the systems I/O.

With the total I/O count completed, it was necessary to propose a hardware setup for the system and to provide the client (FMG) with a price quote. Although modern systems that are being designed for use with the LabVIEW Platform tend to implement the newer Compact Rio hardware, FMG expressed that they already had several Compact Fieldpoint modules in their possession. For this reason, the option of upgrading around the system's existing Compact Fieldpoint hardware had to be considered. Although re-instrumenting the plant entirely with Compact Rio devices would be a desirable option from a performance perspective, the choice of which hardware setup would be implemented was ultimately a decision to be made by Proteus and FMG, based upon their assessment of various factors such as budget, performance, and future compatibility.

In a meeting with representatives from FMG, it was expressed that the hardware used within the plant was to be standardised so that replacement parts could be easily obtained. This standardisation requirement eliminated the possibility of building a system that utilised both the newer Compact Rio devices and the plant's existing Compact Fieldpoint modules. As such, the only two viable options were to either expand the system's existing Compact Fieldpoint devices, or to completely re-instrument the plant with Compact Rio devices. Since Compact Rio devices support both breakout-board (external I/O) and embedded (internal I/O) configurations, it was necessary to consider each of these options independently.

Using the total I/O list generated from the electrical wiring diagrams, spreadsheets were developed in Microsoft Excel to calculate the required number of modules, controllers, expansion chassis, and connector blocks for each available hardware configuration. For the Compact Fieldpoint setup, the parts selected were chosen so that they would be the same model number as the existing hardware. Since the client expressed that it was important for the hardware used to be standardised, any deviation from the existing Compact Fieldpoint hardware would result in the existing configuration being of minimal use, and would likely eliminate the possible cost benefit associated with selecting Compact Fieldpoint over Compact Rio. A price quote was prepared for each hardware configuration, so that the financial factors of each solution could be appropriately assessed by the client.

Detailed pricing spreadsheets for each of the three potential hardware setups can be seen in Appendix 1, Appendix 2, and Appendix 3 for the Compact Rio (without breakout boards), Compact Rio (with breakout boards), and Compact Fieldpoint setups, respectively. The final prices for each hardware setup can be seen in Table 2 below.

Total Hardware Cost	
Hardware Setup	**Cost**
Compact Rio *(Without Breakout-Boards)*	Total System Cost: $20,113.50
Compact Rio *(With Breakout-Boards)*	Total System Cost: $14,668.50
Compact Fieldpoint	Total System Cost: $22,108.90 Upgrade Cost: $8,893.50

Table 2 - Total Hardware Costs

As can be seen from the hardware price-estimates presented in Table 2, upgrading the system's existing Compact Fieldpoint setup works out to be significantly cheaper than re-instrumenting the plant with Compact Rio. Upgrading the current Compact Fieldpoint setup would result in a $5775 initial cost saving over re-instrumenting the plant with breakout-board enabled Compact Rio devices, and an $11,220 initial cost saving over re-instrumenting the plant with embedded-I/O Compact Rio devices.

Although there is a clear short-term financial benefit associated with upgrading the existing Compact Fieldpoint hardware, it is debateable whether this will prove to be the best option in the long-term. FMG has expressed that the Pilot Plant in question is expected to have an operating life of 10-15 years. Since Compact Fieldpoint has reached the end of its development cycle as a product line, compatibility issues may arise with the setup in the future. As time goes on, replacement Compact Fieldpoint parts will likely become increasingly harder to obtain, and the costs of such replacements will likely be higher than the equivalent Compact Rio parts. ICON Technologies believes that, from a performance perspective, the Pilot Plant should ideally be completely re-instrumented with Compact Rio hardware. In ICON Technologies' opinion, an $11,220 difference in initial hardware costs is not worth the trade-off of having the system operate on obsolete hardware for the rest of its lifecycle. Additionally, Compact Rio hardware is more powerful than the existing Compact Fieldpoint hardware, and would allow for more computationally demanding tasks to be performed in the future if required. Although Icon Technologies' recommendation was that the pilot plant be re-instrumented with Compact Rio devices, the decision of which hardware would be used was ultimately a decision to be made by Proteus and FMG.

Following the generation of price-quotes for each possible hardware configuration, a meeting was held with a representative from Proteus to discuss the possible hardware options for the plant. The hardware pricing spreadsheets were presented to Proteus, and ICON Technologies proposed their recommendation of re-instrumenting the plant with Compact Rio devices. After these pricing spreadsheets had been reviewed by Proteus, a revised hardware specification was sent to ICON Technologies, along with a request for an updated price quote. The specification showed that Proteus had taken ICON Technologies' recommendation and had decided to re-instrument the plant with Compact Rio devices (without breakout-boards). However, Proteus chose to leave several blank module spaces in each chassis so that additional modules could be added in the future. They also indicated that they wanted the necessary Power Supplies, Ethernet Switch, and Filler Modules to be included in the price quote.

A revised pricing spreadsheet was constructed, and forwarded to Proteus. The detailed price-quote can be seen in Appendix 3. The total initial hardware cost for Proteus' proposed setup (including GST) came to $27,478.00. At the present moment, the proposed hardware setup has not been purchased owing to delays in the project on the client's end. It has been indicated, however, that this hardware setup will be purchased once the project comes back into operation.

2.2.2 Software Specification

Owing to delays associated with the project's budget, a complete software specification has not yet been developed. However, several meetings have been held with representatives from FMG to discuss the requirements of the HMI that was to be developed under the initial purchase order. Through several preliminary meetings with FMG representatives, the following requirements were expressed:

2.2.2.1 Human Machine Interface (HMI)

- The HMI should be simplistic so that operators can easily follow the process.
- The HMI should not be cluttered, and should make appropriate use of space.
- The HMI should be designed to run full-screen on the client machine and should not require the use of scroll-bars. The client screens will have a resolution of 1920x1200 pixels.
- Hyperlinked navigational buttons should be available on the HMI so that process streams can conveniently be followed across multiple screens.
- The symbols allocated to unit operations should be easily recognisable to operators and should be standardised across the HMI.
- Each unit operation should dynamically change its colour based upon its current operational status. Unit operations that do not have any I/O associated with them should be grayscaled.
    - Green = Operational
    - Red = Stopped
    - Yellow = Error State
    - Purple = Interlocked
- Unit operations for which associated readings can be displayed graphically should operate as such. Specifically, tanks that have an associated level-indicator should have their graphics dynamically changed to correspond to the active level.
- Relevant numerical indicators should be presented for unit operations and important process screens.
- Thick black arrows should be used to denote the flow of process materials, and thick blue arrows should be used to denote the flow of process water.
- Control-loops between system components should be indicated through the use of thin, grey, dotted arrows.
- Each unit operation that is associated with I/O should be selectable by the user. Selected unit operations should be highlighted, and double-clicking on the object should spawn a popup property window.
- Popup windows should display custom control layouts depending upon the selected unit operation and the associated I/O.
- Popup windows should support drag-and-drop interaction and be capable of being repositioned anywhere on the HMI.
- Multiple instances of popup windows should be supported so that the properties of more than one unit operation can be viewed at any given time.
- Proprietary sub-systems of the plant that are externally controlled should be represented by labelled boxes, and should be treated as isolated inputs into the system.

Following the meeting in which these requirements were established, ICON Technologies was provided with three A3 screen-layouts that had been developed by an external contracting company. These screen layouts were constructed for the purpose of outlining the fundamental design of the HMI, as well as to specify which system components were to be displayed on each screen.

After reviewing the provided screen layouts, it became apparent that the designs were not suitable to be directly translated into an HMI layout. Although these designs provided a good indication of the important components of the plant, implementing the HMI as presented on the screen layouts would not prove to be practical. The designs did not appear to take into account the space needing to be occupied by front-panel indicators, and were not designed to the dimensions of the screens specified for the display. Furthermore, the overall flow of the designs resulted in the front-panel components needing to be impractically small if it were to be implemented to scale. FMG was contacted about this issue, and they indicated that ICON Technologies was free to deviate from these designs if a more appropriate solution was available.

2.2.3 Software Development

2.2.3.1 Human Machine Interface (HMI)

2.2.3.1.1 Screen Layouts

The first step towards building an HMI for the Pilot Plant was to establish how many screens were required to be created, and which unit operations would be displayed on each screen. After reviewing P&ID diagrams of the plant, as well as screen-layouts provided by an external company contracting to FMG, it was decided that having three primary HMI screens would be suitable. The screen-layouts presented a good indication of which unit operations should be displayed on each screen. However it was necessary for the location of each unit operation to be re-arranged so that screen-space could be optimised.

To determine the best position for each unit operation, dummy-screens were created that were sized to the same resolution as the client screens (1920x1200 pixels). Rectangular LabVIEW decorations were then created to represent each unit operation, and were positioned and resized on the HMI until a suitable layout was established. Dummy flow-lines were also drawn in so that the overall outline of the HMI could be seen. These dummy screens formed a draft layout for the HMI, and provided an indication of the required size and position of each unit operation and its associated graphic.

2.2.3.1.2 Visual Graphics

After the position and size of each unit operation had been determined, it was necessary to establish workable images for each unit operation. Although LabVIEW's DSC Module provides a variety of inbuilt graphics for many standard system components, they do not exhibit the flexibility and aesthetic appeal that is required for the HMI to be built to the client's specification [1]. Additionally, the DSC Module only provides graphics for a small subset of the unit operations implemented in the pilot-plant, meaning that these graphics would be difficult to standardise. It was decided that custom-building images for each unit operation would be the most appropriate design method.

Images for each unit operation were constructed within external graphical design software, and were carefully sized based upon the dummy HMI screen layouts. Care was taken to ensure that the graphics constructed were simplistic, and would be easily recognisable to plant operators. Each graphic was separately saved in shades of green, red, yellow, and purple so that the colour of each graphic could be dynamically changed within LabVIEW. As an example, the image set designed for a flow-control valve can be seen in Figure 2 below. The state-names are for illustration purposes, and do not form part of the presented graphics.

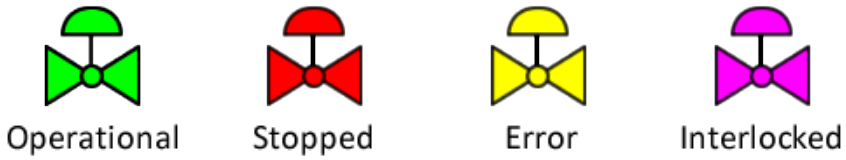

Figure 2 - Flow-Control Valve - Graphical States

2.2.3.1.3 Display Development

With draft screen-layouts developed, and graphics created for each unit operation, it was necessary to physically construct the HMI. The first step in this process was to import each of the constructed graphics into the draft HMI screens so that they could be used as functional unit operations. Since multiple graphics had to be intertwined into single unit operations, graphics could not be directly imported as individual images. Instead, images that formed part of a single unit needed to be imported into different states of a Picture Ring.

A Picture Ring is a form of LabVIEW container that allows multiple images to be contained within a single object [1]. The image that is displayed in the container can be changed programmatically as the program runs. This type of container is ideal for use with HMI images as all of the graphics associated with a single unit operation are contained within the same object. Picture Rings also support image transparency, meaning that they can be used to overlay graphics on top of existing objects.

After the images for each unit operation were appropriately imported into Picture Rings, the rectangular decorations that were used as place-holders in the draft screens were replaced by their respective graphics. With all unit operations correctly positioned on the HMI, it was necessary for dynamic level underlays to be introduced to units that have associated level indicators. Specifically, this allows units such as process tanks to visual display their level.

Since LabVIEW's DSC module contains prebuilt dynamic-level tanks, these tanks were simply used as underlays to the designed HMI graphics. This proved to be a very efficient way of implementing dynamic level indicators, as the filling of customised unit operations did not require any additional coding. Consider the example presented in Figure 3 below, which outlines how dynamic level filling was implemented on the system's Banana Screen.

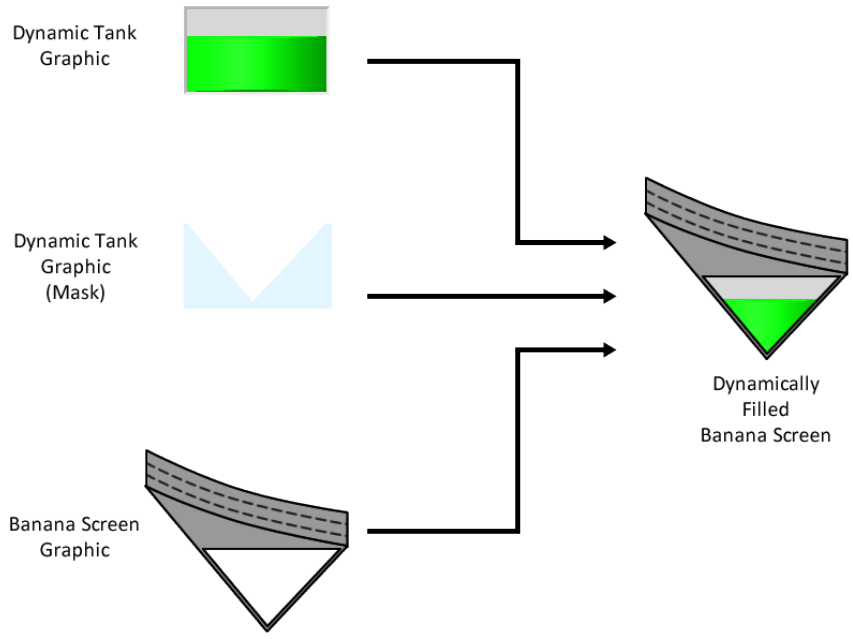

Figure 3 - Creation of Dynamic Level Indicators

After reapplying this process, and adding level indicators to the required unit operations, the dummy-process lines were modified to appropriately match the new graphics. Text labels were also added to the HMI so that the name of each unit operation was clearly specified. Since the HMI has been presented over three separate screens, hyperlinked buttons also were amended to process lines that spanned across multiple screens. This allows process streams to be conveniently followed by the operator.

With the primary layout of the HMI completed, it was necessary to include the appropriate numerical indicators for each system component. The plant's P&ID diagrams were reviewed, and the specified system readings were included on the HMI. Rather than displaying these readings within numerical indicators, system readings were converted to strings and displayed in text boxes. This was for the purpose of amending system readings with the associated units. Once the necessary indicators were added to the HMI, control loops were added through the use of thin, grey, dotted arrows.

After presenting the HMI display to FMG for review, there were a few minor modifications that were required to be completed. These modifications primarily related to incorrect units on some of the system labels, and a discrepancy on the graphics designed for the Tailings Thickener on the second screen. After correcting these issues, the HMI was resubmitted to FMG, who expressed that they were satisfied with the HMI.

Images of the three HMI screens can be seen in Figure 4, Figure 5, and Figure 6 on the following pages. These images can also be seen in Appendix 4.

REMOVED FOR COPYRIGHT PURPOSES

Figure 4 - HMI Screen 1 (Low Resolution)

REMOVED FOR COPYRIGHT PURPOSES

Figure 5 - HMI Screen 2 (Low Resolution)

REMOVED FOR COPYRIGHT PURPOSES

Figure 6 - HMI Screen 3 (Low Resolution)

2.2.3.1.4 Popup-Window Development

With the fundamental graphical component of the HMI completed, it was necessary to create pop-up property windows that would allow system parameters to be configured. In early meetings with the client, the possibility of having a permanently mounted sidebar to interact with selected unit operations was considered. However, it was determined that the sidebar would occupy too much screen space, and that using popup windows would be more efficient.

The first step in developing the popup window interface was to devise a method for determining which object an operator has clicked on. Since LabVIEW does not contain an event structure specifically for clicking on a picture ring, it was necessary to manually code a solution. The approach used is as follows:

1) Upon loading the program, the position of every picture ring is collected relative to the origin (the top-left corner of the screen [0,0]), as well as the picture ring's width and height. This data is bundled in the form [X-Location,Y-Location,X-Location + Width,Y-Location + Height], and is then bundled with name of the object.

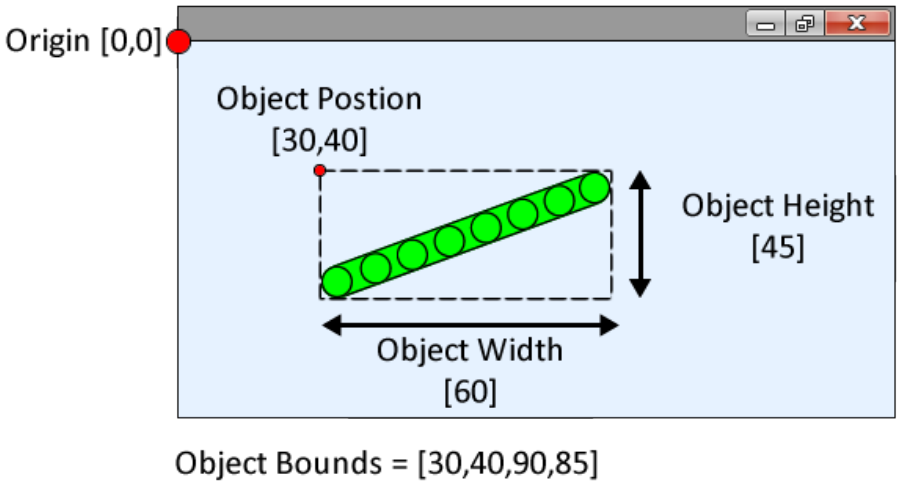

Figure 7 - Calculation of Object Bounds

2) A click event is set up in LabVIEW that returns the current mouse co-ordinates whenever the user clicks the mouse. These mouse co-ordinates are then checked against the data for each object collected in step 1 to see if it is within the bounds of an object.

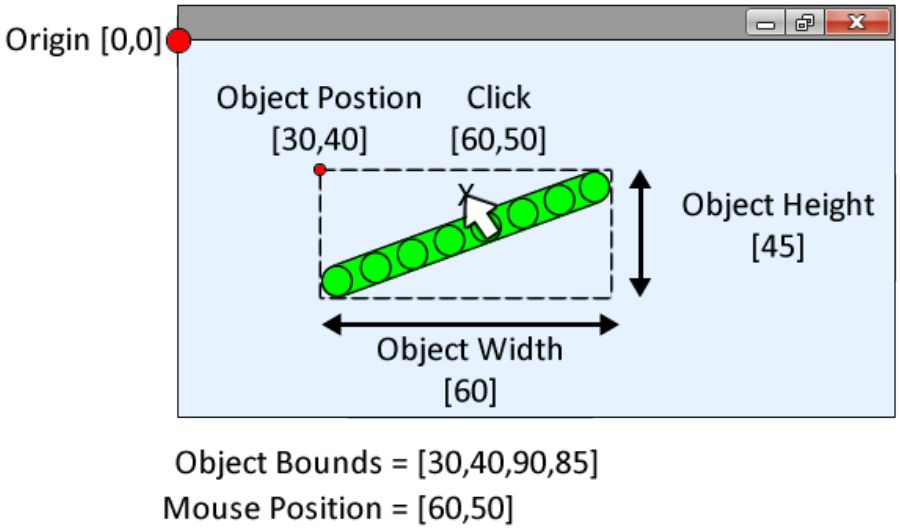

Figure 8 - Analysis of Mouse Co-ordinates

3) If an object is clicked on, the object becomes highlighted through the use of a dynamically sized background image. If an object is clicked on and the user performs a double-click, a popup window will also spawn.

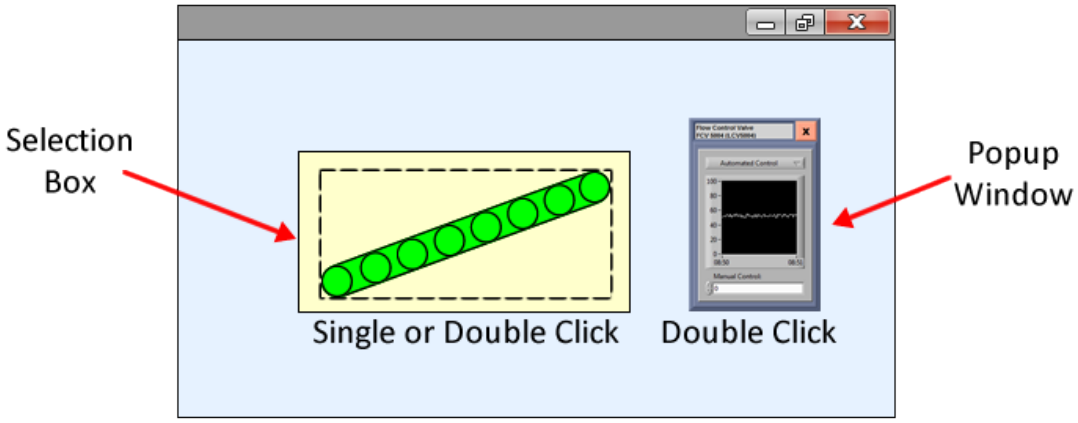

Figure 9 - Active Object Click

It can be noted that the mapping of device co-ordinates only occurs once at the start of the program. This is not a memory-intensive operation, and neither is verifying whether a mouse-click is within the bounds of a mapped unit operation.

The popup windows themselves are constructed so that the contents of the window vary depending upon the type of object selected. Attributes are attached to each object on the HMI that are used to classify them into different types of unit operations. Based on the type of unit operation that has been clicked on, the fields available in the popup window will change. Consider Figure 10 below, which shows example pop-up windows for a Flow Valve, a Flow Control Valve, and a Tank.

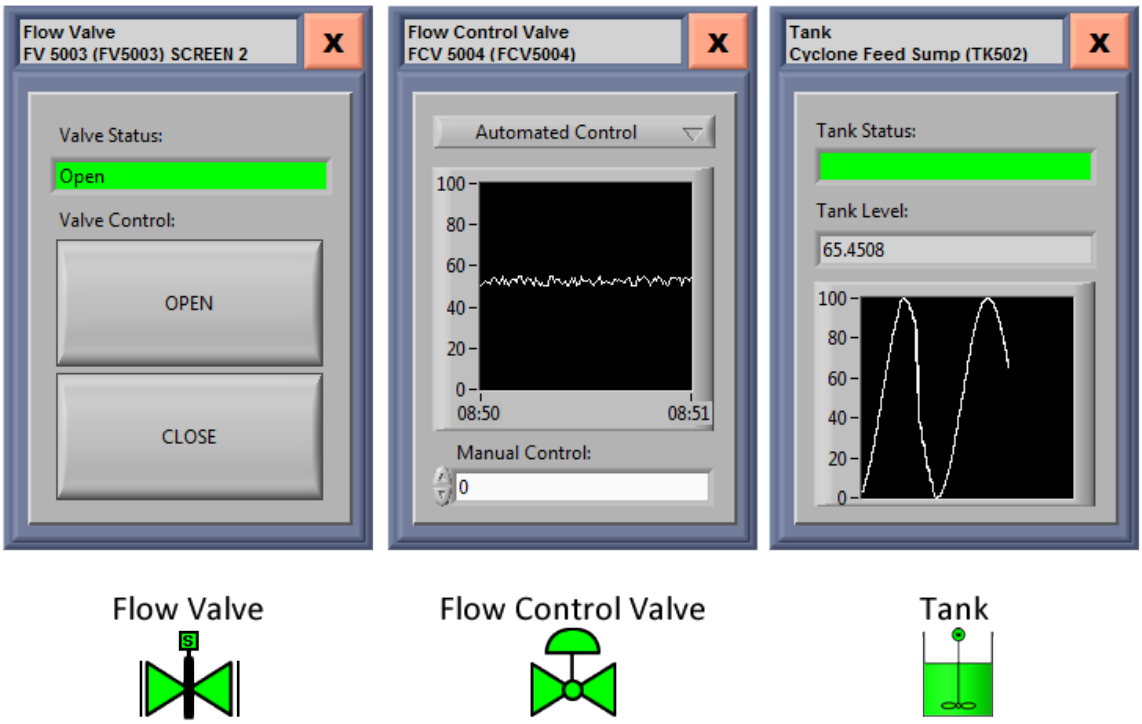

Figure 10 - Example Popup Window Displays

As can be seen, the contents of the popup window differ greatly depending upon the type of unit operation selected. It was determined that applying a default layout to all types of unit operations would not be appropriate because the required controls and indicators vary depending upon the selected object. For example, flow valves only support on and off states, and cannot be set to intermediate values. For this reason, it would not be appropriate to provide the user with the ability to set intermediate valve positions, as this configuration would not be supported by the valve.

The type of object selected, along with the object's name and tag-reference, are clearly displayed at the top of each popup window. Windows can be repositioned on the HMI by dragging them from this display box, and can be closed at any time by clicking on the 'X' in the top right-hand corner. Up to three popup windows can be displayed at any given time. If three windows are displayed, the user will be required to close one before another window can be opened. This constraint was introduced by the client in order to minimise cluttering of the HMI. It is anticipated that alarms associated with each unit operation will also be displayed on these popup windows at a later stage of the project. The configurations for each type of window are only temporary layouts, and are subject to future change.

2.3 Plans for Completion

At the current point in time, the following project tasks still need to be completed:

1) Acquire data from each of the system's inputs, and pass them to the control-system for processing and monitoring. (Data Acquisition)
2) Control outputs from the system based upon user-defined settings, as well as measurements collected from the system through digital and analogue input channels. (Plant Control)
3) Log data relating to the plant's operation. This includes all inputs to the control system, as well as outputs from the control system. (Variable Logging)
4) Recall and review collected data so that it can be post-processed and analysed. (Historical Trending Screen)
5) Automatically generate alarms based upon pre-defined characteristics of system parameters. Appropriate methods for notifying the user of active alarms are also required to be implemented. (Alarm Generation and Monitoring)

Once these tasks have been completed and the control system is functional, the system testing stage needs to take place. After the system testing has been completed, factory acceptance testing (FAT) must be conducted. These factory acceptance tests are then followed by the control system being commissioned.

The project is currently in a position where work can immediately recommence following the creation of a secondary purchase order. However, until FMG is in a position to financially commit to the project, no further work is able to be performed. An estimated time of completion cannot be calculated, since the project has been put on hold indefinitely owing to factors out of ICON Technologies' control.

3 Curtin University – Green Electrical Energy Project

3.1 Project Overview

3.1.1 Project Scope

Curtin University owns a renewable electrical energy plant that, among other purposes, is used for laboratories and workshop classes in relevant engineering courses. Although the components of this system are functional, there is currently no monitoring system in place to allow digital analysis and monitoring of system readings. Curtin is aiming to have a computer-based monitoring system up and running by the first semester of 2012 so that use of the laboratory can be incorporated into the students' course-work. This system is intended to process data as well as monitor it in order that relevant readings can be retrieved by students for use in experiments and system tests.

The university has a large number of field devices that will be required to be interfaced with in a variety of different ways. While a large number of devices will be able to be interfaced with via digital and analogue channels, there is also an external weather station that will need to be interfaced via serial communication, and an SMA Sunny Web Box that will need to be interfaced via Ethernet through the use of a proprietary OPC server. It is required that each of these devices is incorporated into the same monitoring system so that each external device forms part of the same system.

Each input to the system is required to be processed to give readings such as RMS values, FFT's, and other specified calculations such as power factors. Raw data, as well as calculated values based upon this raw data, should be capable of being logged so that long-term trends can be monitored and post-processed. Data should be acquired at a high frequency (5 kHz) on the server so that calculations can be based upon a large number of data points. For system inputs with slow-changing values (such as d.c. battery voltages), readings can be sampled at a much lower rate. After processing, raw measurements can be resampled so that data can be logged and passed to the client programs at an appropriate sample-rate.

Inside the laboratory classroom, there are eight user-stations which can be used to conduct experiments. Since each station will not require the values collected from every input to the system, it is required that subsets of the total system inputs can be shared with each station. The mapping of system inputs to user-stations is required to be configurable on the server. A generic client program is required to be constructed that can be installed on any number of networked computers. The 'Station ID' should be configurable from within the client program so that the client can be applied to any system station. These station clients should have the ability to monitor live data from their associated channels, as well as view historical data that has been logged on the server.

A ‘Master Client’ is also required to be created, which operates the same as a user client but has elevated system privileges. The purpose of this program is to allow the lecturer or person in control of the workshop to view system data collected from each station, and control which variables are being logged and monitored.

In addition to monitoring the system on the internal Curtin University network, it is necessary for a website to be hosted that allows monitoring of the system from outside of the university. This online monitoring system should be simplistic and easy to use so that every-day users will be able to operate it. The data presented on this display is expected to be a cut-down version of the data available from the system so that the general public can gain an understanding of how the lab operates.

3.1.2 Required Tasks

To develop a monitoring system to the client's specifications, there are a number of primary tasks that need to be completed. A brief outline of each task is given below, with a detailed outline of each task being presented in section 3.2.

1) Hardware Specification
   Determine the hardware required for the system, and provide a price-quote to the client.

2) Software Specification
   Discuss each component of the proposed system with the client, and determine the associated constraints, guidelines, and requirements for each section.

3) Temporary Hardware Setup
   Install the hardware in a temporary setup so that it can be used for development purposes.

4) Server Development

    a. Data Acquisition

        i. Compact Rio Devices

            1. FPGA Code
               Develop FPGA code for each Compact Rio chassis that is able to acquire data from each channel of the attached modules at 5kHz.

            2. Real-Time Controller Code
               Develop real-time controller code for each Compact Rio chassis that is able to read data collected from the FPGA, and communicate it over Ethernet to the server program.

            3. Server Code
               Develop code for the server that can retrieve data that has been communicated over the network by the real-time controllers of each chassis.

        ii. Weather Station
            Develop code for the server that can communicate with the Weather Station over an RS232 connection to acquire data at a rate of 1Hz.

        iii. SMA Sunny Web Box
            Install the SMA OPC Server on the server computer, and develop LabVIEW code that can access data from the SMA OPC Server.

b. Channel Mapping

Write code for the server that is able to generate and read XML configuration files. These configuration files shall contain information relating to which system inputs correspond to each client station. The contents of the configuration files will determine which data inputs will be viewable by each client.

c. Data Calculations

Develop code for the server that will process the raw input data, and calculate the RMS and FFT values for each input. Additional client-specified calculations (e.g. Power Factor) are also to be included. Data that was collected at 5kHz will be resampled to 1kHz after it has been processed.

d. Database Logging and Querying

Develop code for the server that is capable of logging all inputs to the system, including calculated values. The server should accept commands from client programs to control logging of their associated variables, and to request database queries. Upon performing a database query, the server should communicate the queried data back to the client who requested it.

5) Station Client Development

Develop a generic client program for the system that can be used on any number of networked computers. The Station ID should be user-configurable, and the client program should retrieve data from the server program based upon this Station ID. Three primary screens should be available on the client display:

a. Process View Screen

Provides a general overview of the system, and displays numerical indications of live system parameters.

b. Live Data View Screen

Provides live data views of system inputs associated with the station. Charts are displayed for the raw data, RMS data, and FFT data of each selected channel.

c. Historical Data / Trending Screen

Provides the ability to query the server's database for historical data, such that long-term trends can be viewed. Data can only be queried for channels associated with the active Station ID.

6) Web Server Development

Develop a web data server that allows the system to be monitored by remote clients outside of Curtin University's internal network. This web-based user interface should be viewable from within a standard web-browser, and present a simplified summary of the system for public access.

7) Commissioning
   The system is installed in Curtin University's desired location, and the system's functionality is tested and verified.

3.1.3 Current Completion Status

At the current point in time, a large amount of the Curtin Green Energy Project has been completed. The system is currently functional within the simulated test setup, and additional features are in the process of being implemented and refined to the client's specifications. The progress upon each major project component outlined in section 3.1.2 is briefly summarized below. (Detailed information relating to the progress made on each of these sections will be provided in section 3.2 - Current Progress.)

1) Hardware Specification
   A hardware specification has been provided to the client, and has been approved. The hardware has subsequently been purchased.

2) Software Specification
   Numerous meetings have been held with representatives from Curtin University to establish a software specification. A clear understanding has been established of what Curtin requires from the software; ICON Technologies has been working closely with Curtin throughout the design process.

3) Temporary Hardware Setup
   The necessary hardware has been purchased and is currently in operation within the testing facility at the office of ICON Technologies.

4) Server Development

    a. Data Acquisition

        i. Compact Rio Devices

            1. FPGA Code
               FPGA code has been developed for each of the system's chassis, and passes readings from each channel input to the associated real-time controller at 5kHz. (Waveforms containing 5000 sample readings are sent to the controller once per second.)

            2. Real-Time Controller Code
               Code has been developed for the each real-time controller to read data from the FPGA and communicate it via Ethernet to the server program. Data is transferred using standard TCP/IP protocols.

            3. Server Code
               The server has been coded to receive data from the real-time controllers via standard TCP/IP protocols. Data is received from each controller as an array of waveforms. Each waveform contains 5000 samples, and is retrieved once per second.

ii. Weather Station

Code has been developed that is able to retrieve the required values from the Weather Stations via an RS232 connection. The LabVIEW code makes use of the 'VantagePro.dll' file provided by the Weather Station's manufacturer for interfacing with the device. Data is collected from the Weather Station by the server once every second (1Hz).

iii. SMA Sunny Web Box

Although the SMA Sunny Web Box has been set up as part of the test installation, data from the device is not yet accessible in LabVIEW. The SMA OPC Server has not yet been purchased by the client, and this is required to reliably interface with the device. Although the Sunny Web Box supports remote procedure calls (RPC) to access data, this method is seen as being too unreliable for use in the monitoring system.

b. Channel Mapping

Code has been developed on the server that is able to generate and read XML configuration files. Configuration files have been generated for the system, and can be modified through the use of an XML editor, or by re-generating them from the designed configuration generator. The server reads the configuration file on start-up, and uses the information contained within the file to map input channels to client stations. The information sent to each client is dependent upon the contents of this configuration file.

c. Data Calculations

The server currently processes each input channel and calculates its RMS value and its FFT. Specific details have not yet been provided as to the additional calculations needing to be performed. Once this information is provided to ICON Technologies, it will be a simple task for the specified equations to be implemented.

d. Database Logging and Querying

The server is capable of logging each input channel to a local Citadel Database. The logging configuration for each variable is set from the respective client program, and is logged by the server accordingly. Each client computer is able to send database queries to the server. The server processes incoming database queries, and returns the retrieved data to the client program that made the request through the use of LabVIEW's Shared Variable Engine (SVE).

5) Station Client Development

A generic client program has been developed for the system that can be used on any-number of networked computers. The Station ID should be user-configurable, and the client program can retrieve live and historical data from the server program based upon this Station ID. Three primary screens should be available on the client display:

a. Process View Screen

A place-holder for the process view screen has been implemented within the client program, but does not yet contain any functional components.

b. Live Data View Screen

The live data screen is currently functional, and allows live data to be viewed for selected channels. The live data screen contains four subpanels, with the first containing a raw data view, the second containing an RMS data view, the third containing an FFT data view, and the fourth containing raw, RMS, and FFT plots on the same display.

c. Historical Data / Trending Screen

The client program is able to send a request to the server to query the database for selected historical data. After the requested query is made by the server, the returned data is sent to the client through the use of LabVIEW's Shared Variable Engine. This information is subsequently displayed on a chart for the user to review and analyse.

6) Web Server Development

The web server component of the system has not yet been implemented.

7) Commissioning

The system is not yet in the commissioning stage. As such, the system has not been commissioned.

3.2 Current Progress

3.2.1 Hardware Specification / Temporary Hardware Setup

Prior to the start of the internship program, a hardware specification and price quote had already been provided to Curtin University by ICON Technologies. Curtin approved the specified hardware setup, and subsequently incorporated this setup into their wiring diagrams for the project. After receiving approval from Curtin, the specified hardware was purchased and delivered to the office of ICON Technologies. It was expressed to Curtin that ICON Technologies would require access to the purchased hardware in order to develop the software. Subsequently, it was decided that a temporary hardware setup should be established at ICON Technologies so that code could be developed for each device.

Upon receiving the required hardware from National Instruments, it was necessary to read through the wiring diagrams provided by Curtin to appropriately set up the hardware. From these wiring diagrams, an I/O List was developed for the purpose of verifying the modules needing to be installed in each chassis. This I/O List can be seen in Appendix 5. After the finalised hardware setup was verified from the wiring diagrams, the required modules were installed in each chassis. Spare module slots were filled with blank filler-modules so that exposed connectors would not become damaged. Each chassis was then set up at a testing station and connected to ICON Technologies' internal network via an Ethernet connection. A photograph of the Compact-Rio hardware setup can be seen in Figure 11 below.

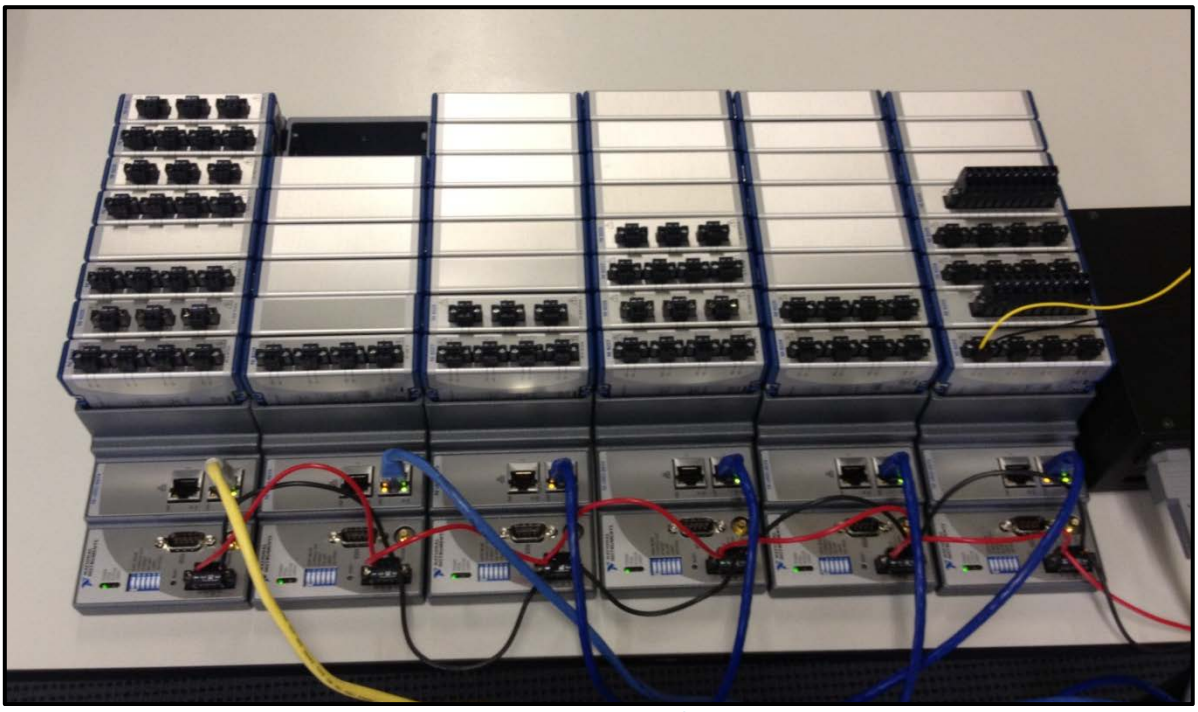

Figure 11 - Compact Rio Hardware Setup

In addition to setting up the system's Compact Rio devices, it was also necessary to set up the additional system components that were required to be interfaced with. Specifically, this included a Vantage-Pro Weather Station, an SMA Windy Boy Inverter, and an SMA Sunny Web Box. The weather station was set up at the testing station at ICON's office, and was connected to the server computer via an RS232 connection. A small fan was set up near the weather station to enable different weather conditions to be simulated for testing purposes. Photographs of the temporary weather station setup and the weather station's inbuilt operator panel can be seen in Figure 12 and Figure 13 respectively.

Figure 12 – Vantage Pro Weather Station (Temporary Setup)

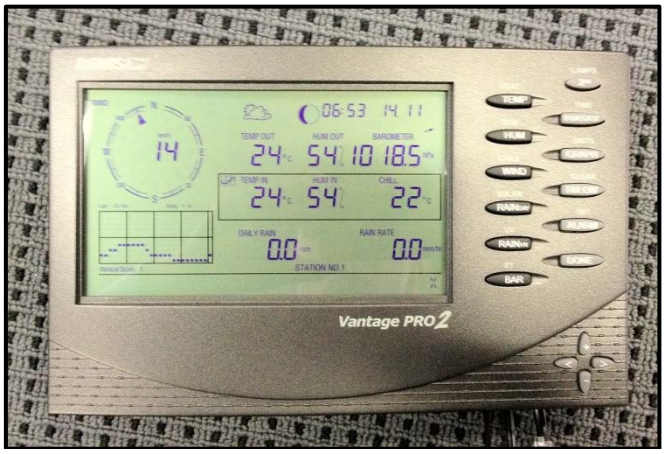

Figure 13 – Vantage Pro Weather Station [Operator Panel]

The SMA Sunny Web Box and Windy Boy Inverter were also set up in the testing facility. Since the Sunny Web Box acts as a gateway to the Windy Boy, the Windy boy was attached to the Sunny Web Box via an RS485 connection. The Sunny Web Box was then connected to ICON Technologies' internal Ethernet network to allow remote access to the inverter. Photographs of the temporary Sunny Web Box setup and Windy Boy Inverter setup can be seen in Figure 14 and Figure 15 respectively.

Figure 14 - SMA Sunny Web Box (Temporary Setup)

Figure 15 - SMA Windy Boy Inverter (Temporary Setup)

3.2.2 Software Development

The software development stage of the project has involved the construction of two primary pieces of software. The first primary piece of software that has been developed is the server program. The server program is responsible for completing the following fundamental tasks:

- Data Acquisition
- Channel Mapping
- Data Calculations
- Client Communications
- Data Logging
- Database Querying

The second primary piece of software that has been developed is the client program. The client program is responsible for completing the following fundamental tasks:

- Retrieving Live Data (from the server)
- Requesting Historical Data (from the server)
- Displaying Data
- Configuring Server Parameters

The progress made on both the server program, and the client program is presented within the following pages.

3.2.2.1 Server Development

3.2.2.1.1 Data Acquisition

One of the key functions of the server program is to routinely collect data from field devices attached to the system. At the current point in time, data is collected from three primary types of devices. These devices include seven individual Compact Rio chassis, a Vantage Pro 2 Weather Station, an SMA Sunny Web Box, and an SMA Windy Boy inverter. The data acquisition systems used by the server to collect data from each of these devices is presented below.

3.2.2.1.1.1 Compact Rio Devices

To acquire data from each of the Compact Rio devices, there are three primary levels of code that have been developed. The lowest level of code resides on the field-programmable gate array (FPGA) of each chassis, and is responsible for the high-speed collection of data from each of the attached modules. Real-time code has been developed for the controller of each chassis that retrieves data collected by the FPGA, and communicates it to the server program. Code has subsequently been developed within the server program to support the retrieval of this data. Details relating to the FPGA code and the real-time controller code used for data-acquisition from the Compact Rio devices are presented below.

3.2.2.1.1.1.1 FPGA Code

FPGA code has been developed for each chassis that periodically acquires data from the device's input channels. Each chassis can support up to eight input modules, with information relating the device's module configuration needing to be hardwired into the code. Although the general process used for acquiring data into the FPGA can be generically applied to each chassis, separate FPGA code needed to be developed for each chassis owing to variations in module configuration.

Data is acquired from each input channel to the device at a rate of 5kHz. This sample rate ensures that sufficient data will be available to perform calculations upon high-speed waveforms, and that sufficient resolution will be available on the Fast Fourier Transform (FFT) plots. Although data can be acquired directly from the real-time controller without using the FPGA, the real-time controller is not capable of acquiring data at a rate of 5kHz. Owing to the high-speed acquisition requirements of the system, it is necessary for data acquisition to be performed on the FPGA.

3.2.2.1.1.1.2 Real-Time Controller Code

Real-time controller code has been developed for each chassis that is able to collect data that has been acquired through the FPGA. The real-time controller typecasts the data collected from each channel into a waveform, and subsequently bundles each channel's waveform data to create an array of waveforms. A TCP/IP connection is established with the server by each chassis, and data packets are sent to the server once per second. Each data packet contains an array of waveforms, with each waveform containing 5000 individual data points.

A buffer has been created in the real-time controller code that allows packages to be stored until they are able to be read by the server. If packages are delayed, they are sent to the server at the fastest possible rate so that the system can promptly catch up to processing the most recently acquired data. A buffer limit of 10 elements has been set so that, if there are more than 10 waveform arrays in the queue, the oldest data is discarded. The buffer has been implemented as a method of minimising gaps in the data whilst the server is operational, but lags for a minimal amount of time. The buffer is only intended to protect against short-term server communication loss, and not act as a backup mechanism for when the server is offline.

3.2.2.1.1.2 Weather Station

To acquire data from the Vantage-Pro 2 weather station, the device has been connected to the server computer via an RS232 Connection. A well-documented .dll file was provided for the device by the manufacturer, which included a variety of functions for acquiring values from the weather station. This meant that raw serial commands were not required to be sent to the device, as data could be requested by simply referencing to the .dll file within LabVIEW.

Function-call blocks were used within LabVIEW to request the required data from the weather station at a rate of 1Hz. A low sample rate of 1Hz was set since all values collected by the weather-station are expected to be slow-moving values. It is expected that data will be requested by the client at a much slower update rate than 1Hz. Since each function block acquires a single system reading, it was necessary to typecast each reading into a single-point waveform, and bundle each reading with the same timestamp into an array of single-point waveforms. This conversion is to allow data collected from the weather station to be processed by the program in a similar way to the data collected from the Compact Rio devices.

3.2.2.1.1.3 SMA Sunny Web Box / Windy Boy Inverter

To acquire data from the SMA Windy Boy Inverter, the device has been connected to the Sunny Web Box over an RS485 connection. The Sunny Web Box acts as a gateway to the inverter, and allows it to be interfaced with via an Ethernet connection rather than an RS485 Connection. There are two primary methods for interfacing with the SMA Web Box within LabVIEW. The first method is through the use of Remote Procedure Calls (RPC), and the second method is through the use of SMA's proprietary OPC server. Although interfacing with the device over RPC would not require the purchase of SMA's OPC server, this is a less desirable option for communicating with the device. The OPC server would likely provide a more reliable method for interfacing with the device, and the client has opted to purchase SMA's OPC server. At the current point in time, the purchase of this OPC server software is still in progress, and no data is yet able to be acquired from the inverter.

3.2.2.1.2 Channel Mapping

Channel mapping code has been developed on the server to map channel readings to their corresponding stations. Three primary sub-VIs are used by the server program to complete this process. The purpose and functionality of these three sub-VIs is presented below.

3.2.2.1.2.1 "Map Channel to Data.vi"

The first sub-VI used is titled "Map Channel to Data.vi", and is responsible for defining which modules are in each slot of each chassis. Configuration data for each chassis is hardwired into the code, since changes to this data will not add any functionality without re-coding the FPGA. Consider Figure 16 below, which shows the configuration data for each chassis.

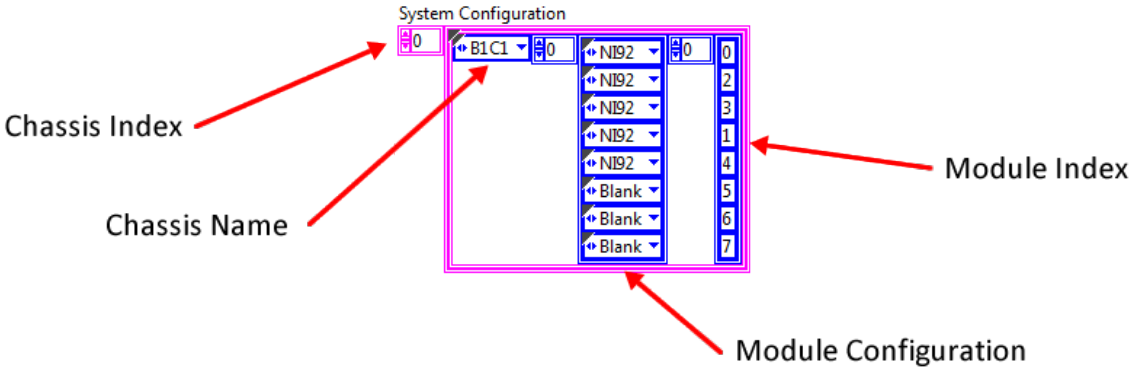

Figure 16 - Chassis Configuration Data

As can be seen from Figure 16, an array of clusters is hardwired into the program to specify the configuration of each chassis. One array element is created for each chassis available on the system, and includes a list of modules installed in each slot, as well as an index number for each module. A secondary array of clusters has also been hardwired into the sub-VI that specifies the parameters of each module. Specifically, each type of module is linked to a string that represents each channel's units, and an integer that represents the number of channels available on the module. This array of clusters can be seen in Figure 17 below.

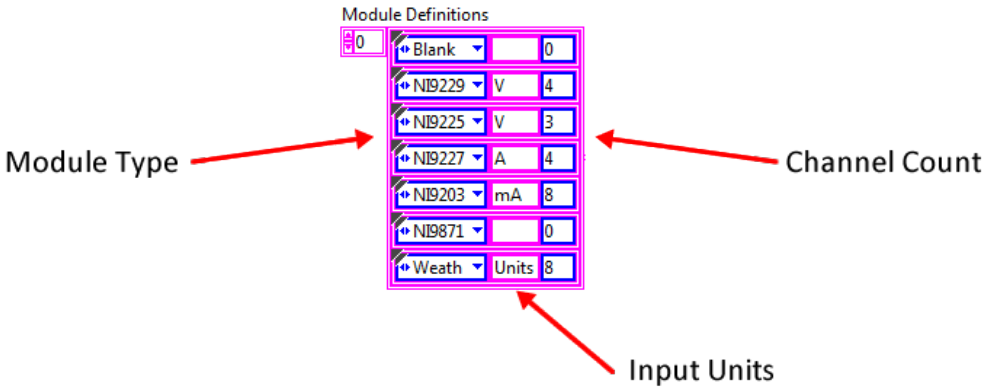

Figure 17 - Module Configuration Data

3.2.2.1.2.2 "Get Station Indexes.vi"

The second sub-VI used for channel mapping is titled "Get Station Indexes.vi", and is responsible for mapping each input channel to a tag name. Default configuration data for each channel's logging settings is also included. Ideally, this program is only designed to be run once, upon initially installing the program, to set an initial configuration for channel names. After the initial channel mapping process has been completed, an XML file is generated by "WriteConfigSvr.vi" that contains these settings. This file is read by the server program on start-up, meaning that any channel re-naming can be performed by appropriately editing the XML configuration file. At the current point in time, there is no graphical configuration utility for modifying channel names. However, a graphical configuration window can be designed if it is later requested by the client.

3.2.2.1.2.3 "WriteConfigSvr.vi"

The third and final sub-VI used in the channel mapping process is titled "WriteConfigSvr.vi", and is responsible for collecting configurations generated by "Map Channel to Data.vi" and "Get Station Indexes.vi". Each channel that has been processed by these sub-VI's is mapped to a specific user station, such that channel readings are appropriately linked to each station based upon the experiment being performed. At the current point in time, these channel mappings are hardwired into the code. However, a graphical mapping utility can be designed if requested by the client.

Once each channel has been mapped, an XML configuration file is written to a specified location on the server. Upon running the server program, the server checks for a configuration file in this specified location. If a configuration file is found, that configuration will be loaded and the mapping process described above will not be performed. The process that has been described in the 'Channel Mapping' section will only be implemented if no configuration file is found upon running the server program.

An extract from the generated XML configuration file can be seen in Figure 18 below. An example of the channel mapping configuration is highlighted in green.

```
<?xml version='1.0' standalone='yes' ?>
<GXML_Root>
 <Sever_Config mems='4'>
  <Port type='U16'>6340</Port>
  <Database_URL type='Path'>\\localhost\Default_Database</Database_URL>
  <Channel_Config dim='[52]' type='Cluster'>
   <Channel_Spec mems='7'>
    <Channel_Name type='String'>TS1_SHNT1</Channel_Name>
    <Chassis type='Enum U16' sel='B1C1'>0</Chassis>
    <Slot type='Enum U16' sel='Mod1'>0</Slot>
    <Channel type='I32'>0</Channel>
    <Station type='I32'>1</Station>
    <lograw type='Bool'>FALSE</lograw>
    <logrms type='Bool'>TRUE</logrms>
```

Figure 18 - XML Configuration File (Extract)

3.2.2.1.3 Data Calculations

The server has been configured to perform various calculations upon raw data acquired from the system. At the current point in time, Root Mean Square (RMS) calculations and Fast Fourier Transform (FFT) calculations are performed upon each raw data input. Additional calculations, such as those required to determine Power Factor (PF), shall be included in the program once the client specifies a detailed set of calculations that need to be performed.

The code for performing data calculations has been encapsulated in a sub-VI. This sub-VI has been named “Process Data.vi”, and is called continuously by the server program. Since new data is acquired into the system once per second, the processing loop typically calls this sub-VI once per second. However, since the processing loop is fed data from the data acquisition buffer, this loop can process data at a faster rate if there are elements queued in the buffer.

LabVIEW code for the “Process Data.vi” sub-VI can be seen in Figure 19 below. This code is presented for the purpose of demonstrating the overall structure of the code, and is not intended to be presented in a format that will allow specific elements of code to be understood.

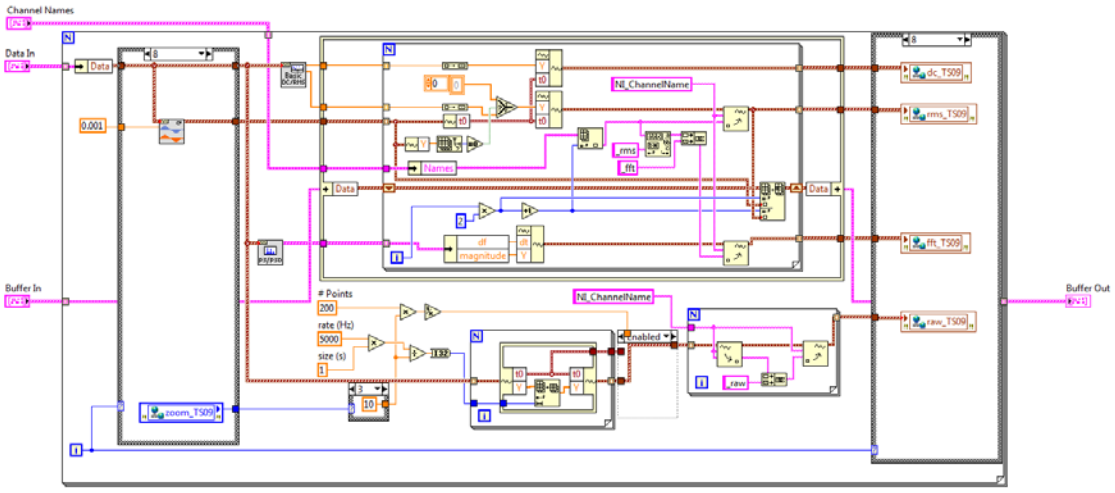

Figure 19 - LabVIEW code for "Process Data.vi"

The code used to perform RMS and FFT calculations is relatively simplistic, since LabVIEW’s ‘Waveform Measurement’ palette contains inbuilt blocks to perform these operations. Data is input into the sub-VI as an array of clusters, with each of these clusters containing an array of waveforms. Each channel’s data is in indexed out of the input data, and processed through the RMS blocks and FFT blocks to perform the necessary calculations. Channel Names are fed into the sub-VI as an array of clusters, with each of these clusters containing an array of strings. Since the indexing levels are the same for each combination of channel data and channel name, channel names can easily be added to the attributes of the calculated waveform outputs. Channel names are amended by “_rms” and “_fft” for RMS and FFT waveform outputs respectively. Once calculations have been performed on the data from each input channel, output waveforms are bundled into an array of waveforms for each specified Station ID. Each array of waveforms is then written to a corresponding shared variable so that it can be accessed in multiple locations within the server program. The “Process Data.vi” sub-VI is also used to collect subsets of the raw data to correspond to requests from the client program. This process is mentioned in section 3.2.2.2 - Station Client Development.

3.2.2.1.4 Database Logging and Querying

Database logging and querying is a key function of the server program, as it allows system data to be stored, reviewed, and post-processed. The processes used for logging into the Citadel database, and for querying from the Citadel database are documented below.

3.2.2.1.4.1 Database Logging

The server has been programmed such that it is capable of logging values associated with the system to corresponding traces in the Citadel database. This encompasses the logging of raw data collected from the system, as well as the logging of processed data that has been calculated by the server. The database that stores system traces is hosted on the server, and cannot be written to by any computer other than the local host.

The decision to limit access to the database was made for two primary reasons. The first reason was that potential security issues exist with allowing multiple computers to write to the database. Only allowing the server to write to the database ensures that existing historical data cannot be over-written without direct access to the server. The second reason for limiting database write access to the server computer was that allowing multiple computers to write to the database would require additional licences to be purchased by the client for use of LabVIEW's DSC module. This is undesirable, as it would increase both the initial costs, and operating costs associated with the monitoring system.

Logging has been implemented on the server program through the use of two primary sub-VIs. The first sub-VI used is titled "Open Traces.vi", and is responsible for initiating trace data for each system value, as well as providing a reference to each trace. The second sub-VI used is titled "Write Traces.vi", and is responsible for logging collected data to the database. The fundamental operation of each of these sub-VIs is described within the following pages.

3.2.2.1.4.1.1 "Open Traces.vi"

The primary purpose of the "Open Traces.vi" sub-VI is to initiate traces for each system value, and to provide a reference to each trace. Consider the code contained within the "Open Traces.vi" sub-VI that is presented in Figure 20.

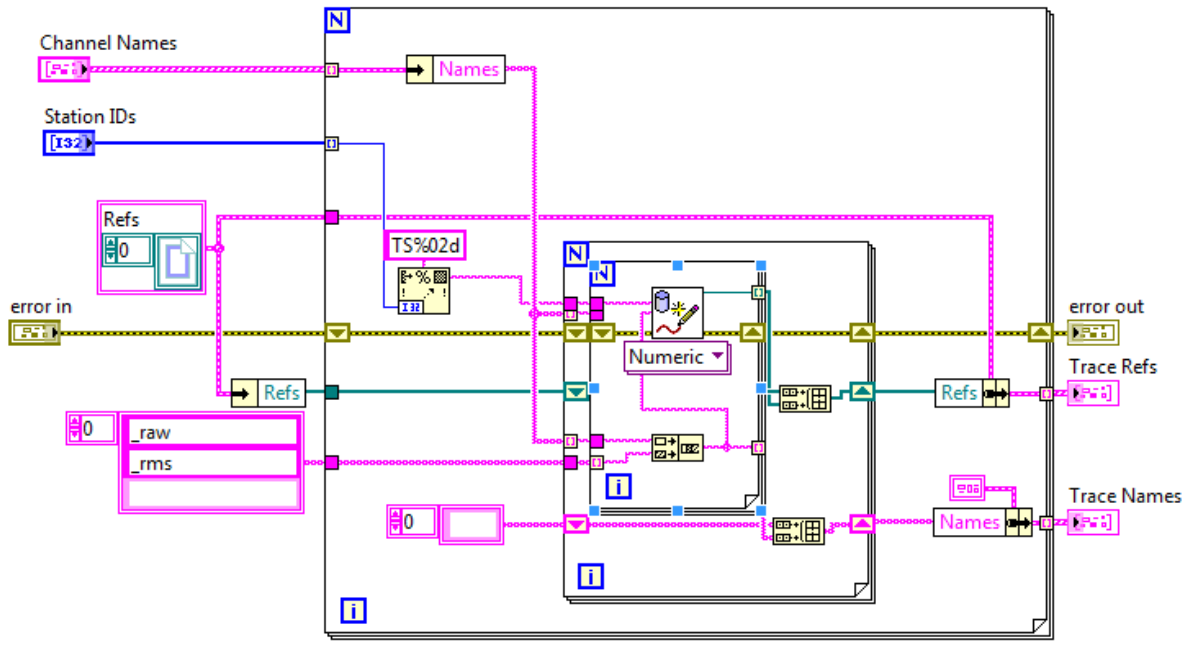

Figure 20 - "Open Traces.vi" LabVIEW Code

Channel names are passed into the sub-VI as an array of clusters, with each cluster containing a string array of names. Each array element in the outer array is associated with a Station ID, meaning that both to channel names array and the Station ID array can be indexed within a FOR loop. Trace names are generated for each trace based upon the indexed Station ID, and the indexed channel name. Traces are created for both raw and RMS values, with trace names being amended with "_raw" and "_rms" for raw data and RMS data respectively.

Traces are initiated using the 'Open Traces' block available from within the 'Database Writing' palette of the DSC module. Inputting the specified trace name into the block opens the trace, and allows it to be written to at a future point in time. An array of clusters is output from the sub-VI, with each cluster containing an array of references to the corresponding traces. A secondary array of clusters is output from the sub-VI, with each cluster containing a string array of corresponding trace names.

3.2.2.1.4.1.2 *"Write Traces.vi"*

The primary purpose of the "Write Traces.vi" sub-VI is to allow selected traces to be actively written to the database. Consider the code for the "Write Traces.vi" sub-VI presented in Figure 21 below.

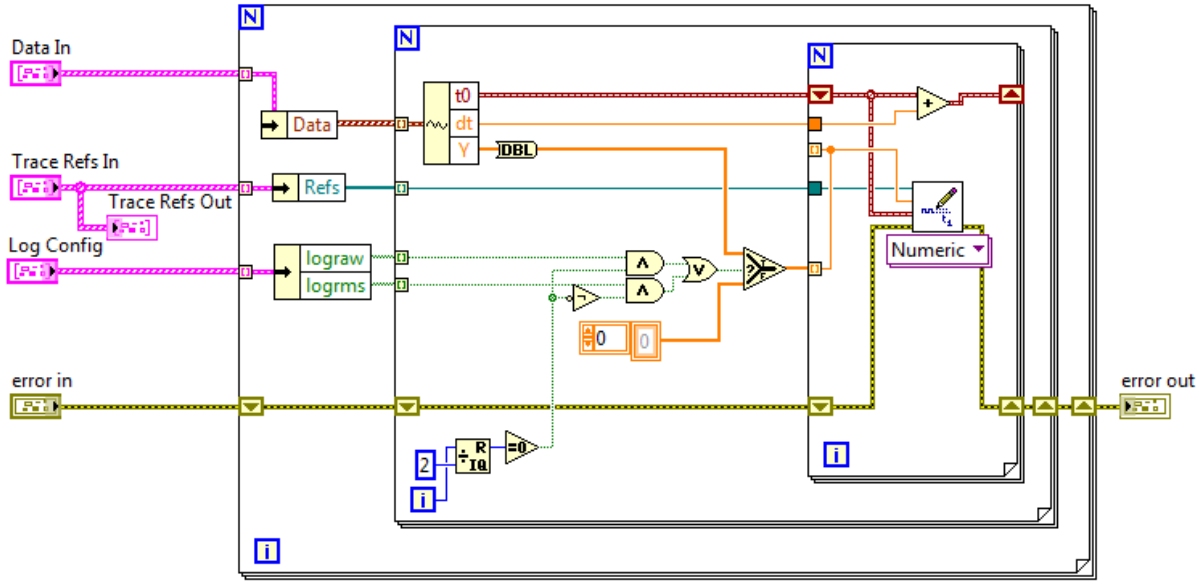

Figure 21 - "Write Traces.vi" LabVIEW Code

The "Write Traces.vi" sub-VI is periodically called from the server program to log data to the database. Data is input to the system in the form of an array of clusters, with each cluster containing an array of waveforms. Logging configuration, as well as a reference to each trace is input to the sub-VI such that each data channel will have a corresponding reference and logging configuration when indexed by a series of FOR-loops. The outer FOR-loop indexes the data by Station ID, the middle FOR-loop indexes the data by station channel, and the inner FOR-loop indexes each channel's waveform data by data-point. Each channel's waveform data is split up into its individual components such that it can be written to a trace. The logging configuration for each data channel determines whether or not the channel is written to the database. If logging is turned off for a given channel, an empty array is fed into the inner FOR-loop, resulting on it having a run count of zero (the run count of an unwired FOR-loop is controlled by the size of its smallest indexed terminal).

3.2.2.1.4.2 Database Querying

The server has been programmed so that it is able to perform queries to the Citadel database based upon requests made by client programs. Each client program can specify a request to the server through the use of the Shared Variable Engine. To specify a request, the client needs to input the following data into the front-panel of the client program:

- A start timestamp for the request.
- An end timestamp for the request.
- An interpolation interval for the data to be returned at.
- A selection of traces needing to be queried.

Detailed information relating to the client program's input to the query will be provided in section 3.2.2.2 - Station Client Development.

Once the user submits a request from the client program, the data is bundled into a cluster. This cluster contains a cluster of timestamps to specify the start and end time of the query, a numerical indicator to specify the interpolation interval of the query, and a string array of trace names to specify the traces needing to be queried. This cluster is then bundled into a shared variable associated with the client's Station ID. A Boolean flag value contained within the shared variable library is then set to true, such that the server knows to read the request cluster written by the client. This cluster is then interpreted by the server, and subsequently used to query Citadel for the requested data. Once the data is retrieved from the database, the server converts it to an array of waveforms and typecasts this array to a variant. This variant is then written to the shared variable engine in the client's associated Citadel-request variable, and the Boolean flag is changed back to false such that the client is notified the server has written the required data to a shared variable. The client program is then able to interpret this data and appropriately display it to the user.

3.2.2.2 Station Client Development

At the current point in time, a functional station client has been created to interface with the server program. The station client is generic, such that it can be used on any-number of networked computers. The Station ID is user-configurable, and the client program is able to retrieve data from the server program based upon this Station ID. It was specified by the client that three primary screens were required to be displayed to the user; a 'Process View' screen, a 'Live Data' screen, and a 'Historical View' screen. The Process View screen has not yet been implemented, as specific information relating to its construction has not yet been provided by the client. The Live Data screen and the Historical Data View screen are currently functional. Details relating to the Live Data screen and the Historical Data View screen are presented within the following pages.

3.2.2.2.1 Live Data Screen

The Live Data screen contains four primary tabs to display information to the user. The first tab allows the user to view raw data, the second tab allows the user to view FFT data, the third tab allows the user to view RMS data, and the fourth tab allows users to view all three of these charts on a single screen display. The construction of each of these screens is outlined below.

3.2.2.2.1.1 Raw Data Screen

The Raw Data screen allows raw data collected from each of the mapped station channels to be viewed by the client. A screenshot of the Raw Data screen is presented in Figure 22 below.

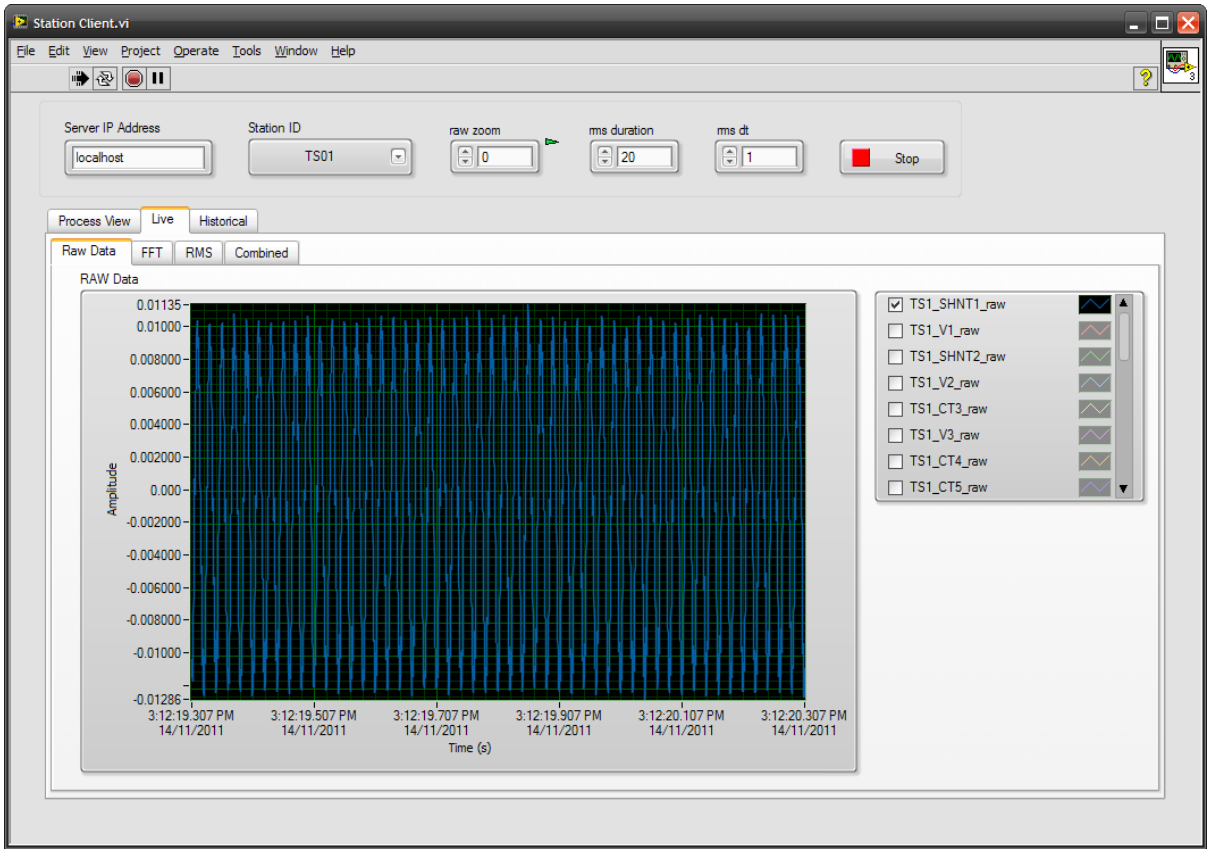

Figure 22 - Raw Data View Screen

As can be seen from Figure 22, each available data channel is displayed in the legend to the right of the chart. Channels can be toggled on and off by clicking the checkbox to the left of the channel name. Data on this screen is updated once per second, and displays data in one second chunks. Since raw data is acquired from each Compact Rio channel at a rate of 5kHz, 5000 data points are presented for each one second chunk of data.

A 'raw zoom' control has been made available for the raw data tab that allows a subset of this raw data to be displayed. The raw zoom control can be set to 0, 1, 2, 3, or 4, with each value being associated with a different zoom level. The characteristics of each of these five zoom levels are as follows:

Zoom = 0: 1 second of data with 5000 data points.
Zoom = 1: 500 milliseconds of data with 2500 data points.
Zoom = 2: 200 milliseconds of data with 1000 data points.
Zoom = 3: 100 milliseconds of data with 500 data points.
Zoom = 4: 50 milliseconds of data with 250 data points.

By setting a zoom level other than 0, a subset of the complete 5000 point waveform is displayed each second. There is no resampling involved, and data on the chart is still updated once per second. Data outside of the time-span specified by the zoom level is simply not displayed to the user. For example, consider Figure 23 below that shows a raw data chart with a zoom level of 4.

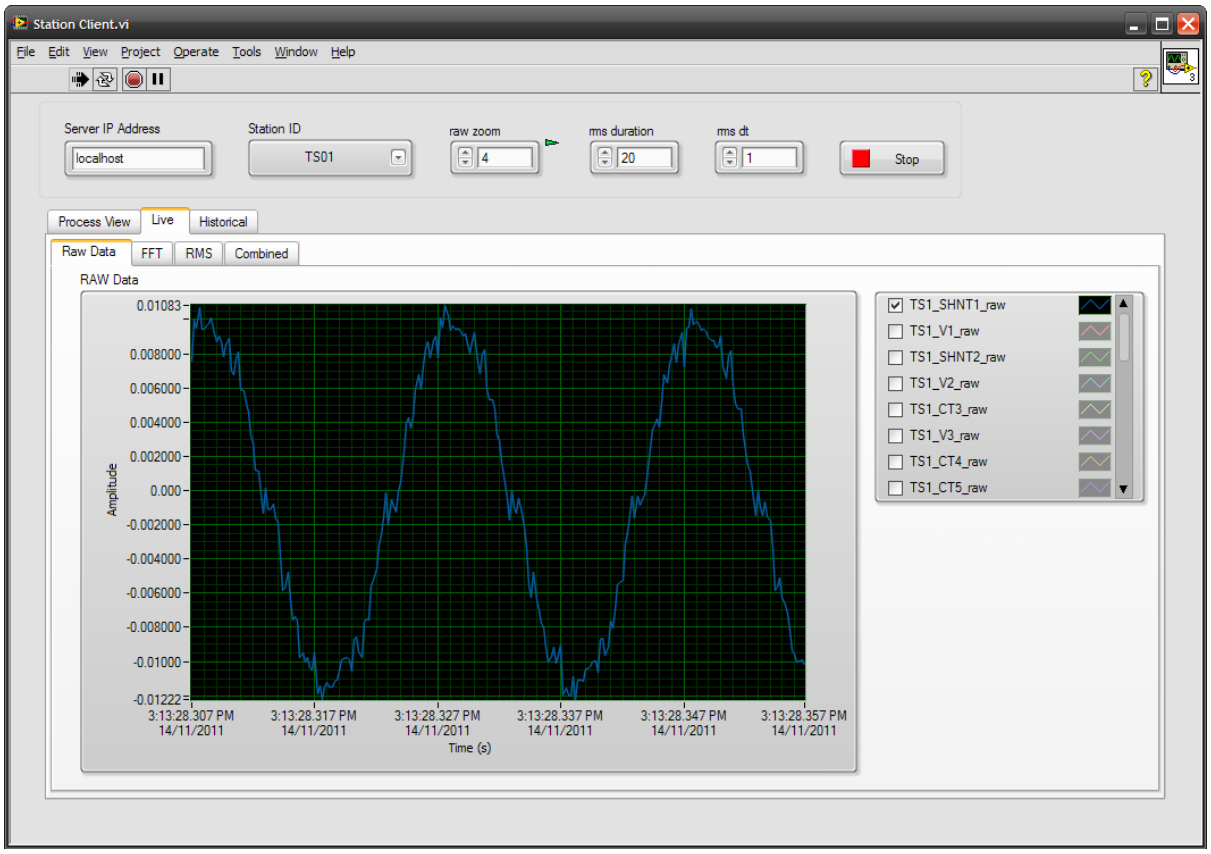

Figure 23 - Raw Data View Screen (Zoom = 4)

As can be seen from Figure 23, only 50 milliseconds worth of data is displayed on the plot. This allows the raw data's characteristics to be more clearly seen. For every one second data chunk received by the client, the first 50 milliseconds will be displayed, and the final 950 milliseconds will be discarded.

3.2.2.2.1.2 FFT Screen

The Fast Fourier Transform (FFT) screen provides an FFT chart for each selected channel. This allows the different frequencies to be seen that make up the input signal. Similar to the Raw Data screen, the visibility of available channels can be selected from the plot legend. A screenshot of the FFT screen is presented in Figure 24 below.

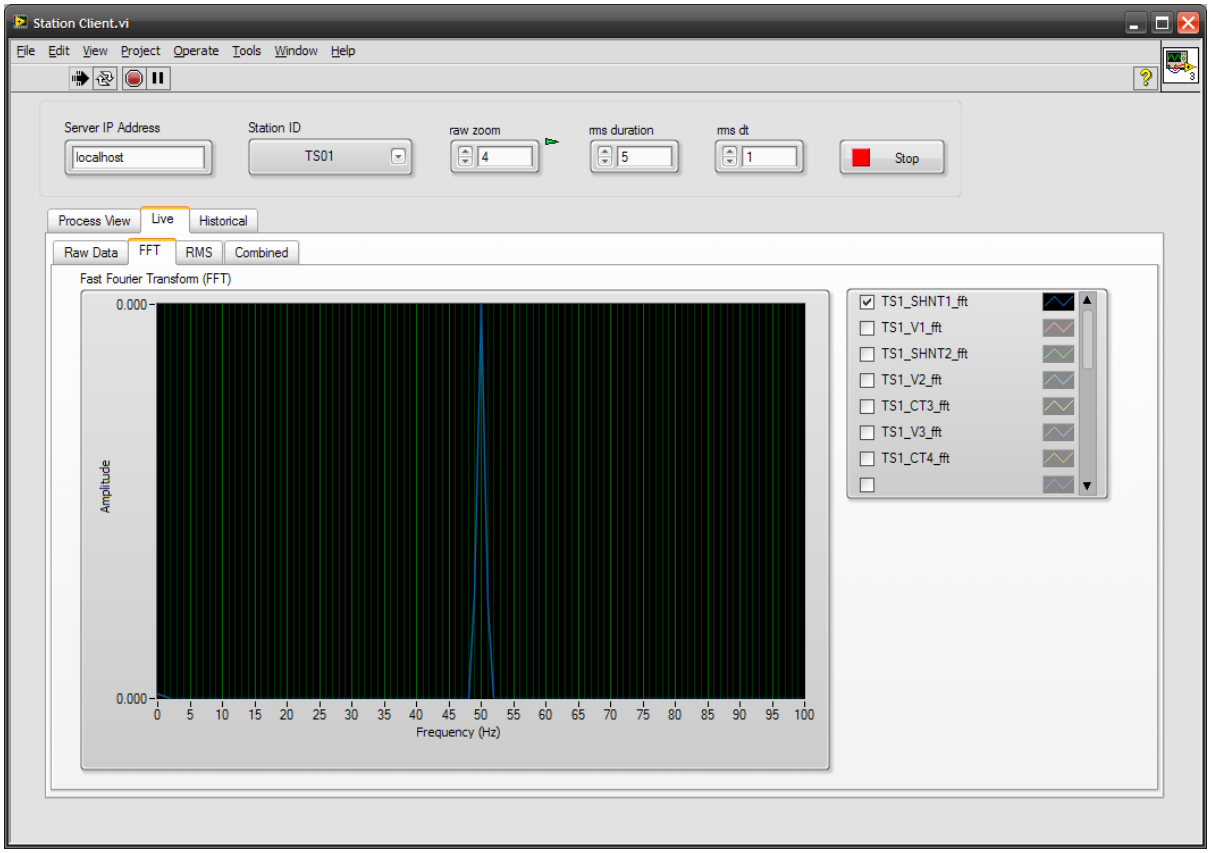

Figure 24 - FFT View Screen

3.2.2.2.1.3 RMS Screen

The RMS screen allows the RMS values calculated for each of the mapped station channels to be viewed by the client. A screenshot of the RMS screen can be seen in below.

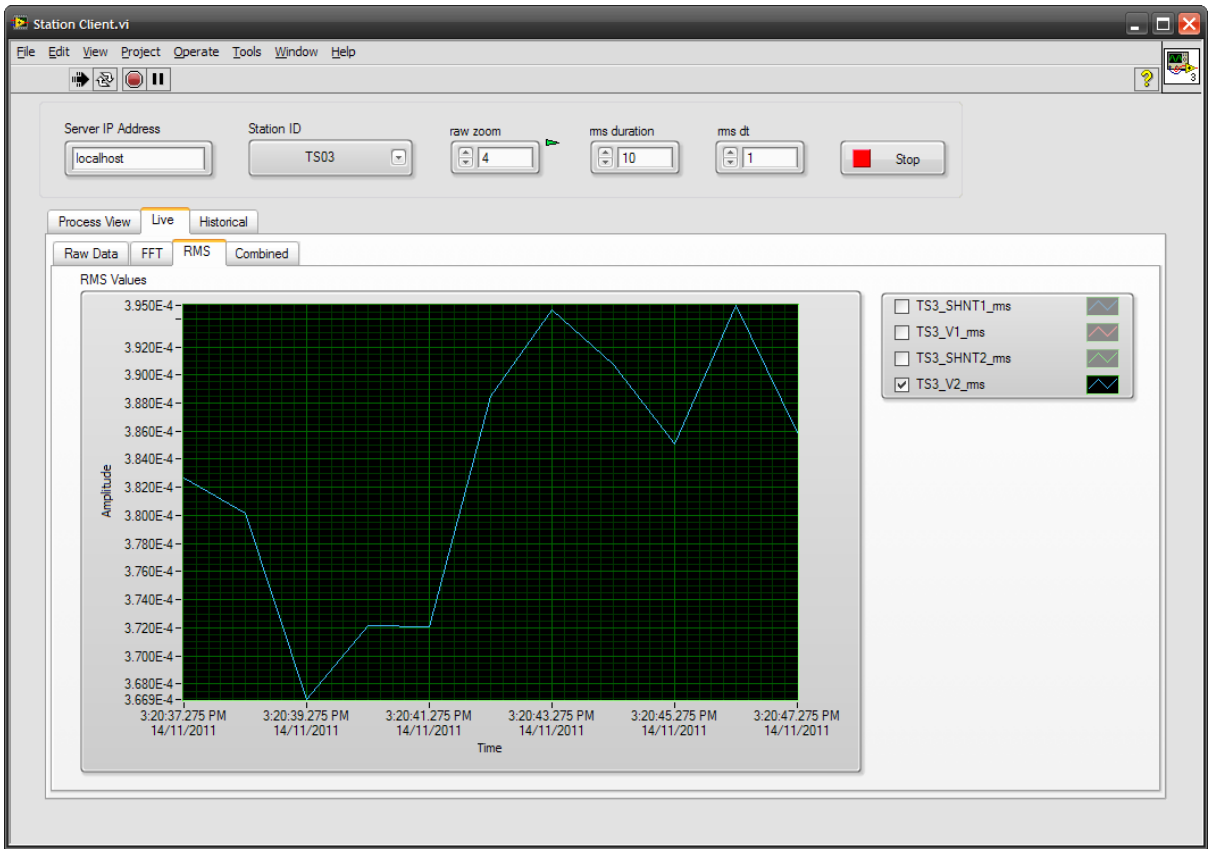

Figure 25 - RMS Screen (Duration = 10s, dt = 1s)

There are two configurable parameters for the RMS chart that enables the user to modify how data is requested from the server. The 'rms duration' control allows the timespan of the requested data to be selected. Setting 'rms duration' to 10 seconds means that 10 seconds worth of data will be displayed on the chart. The 'rms dt' control allows the sample time of the collected data to be selected. Setting 'rms dt' to 1 second means that 1 data point is provided every second. From these two settings, it can be seen that the chart shown in Figure 25 consists of 11 data points.

Consider Figure 26 that is presented on the following page.

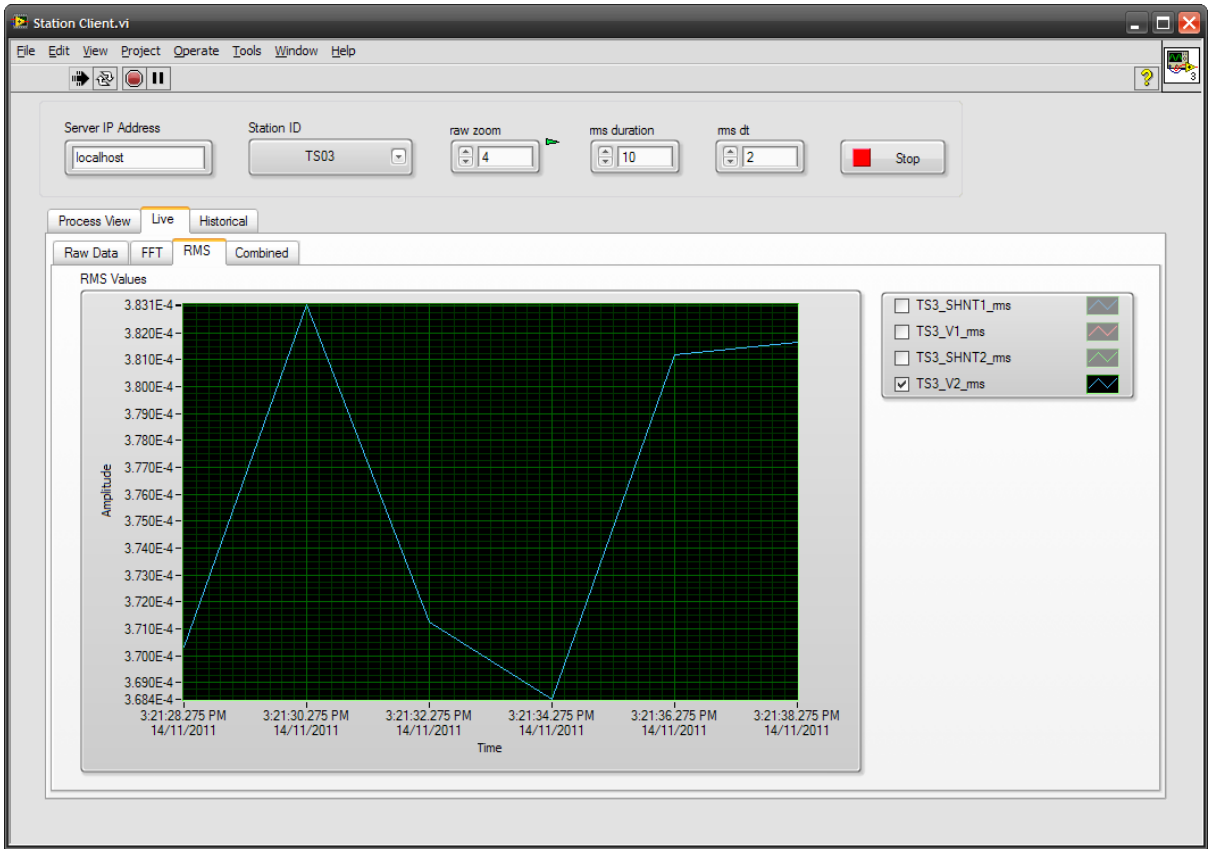

Figure 26 - RMS Screen (Duration = 10s, dt = 2s)

The plot shown in Figure 26 above features an ‘rms duration’ of 10 seconds, and an ‘rms dt’ of 2 seconds. This means that 10 seconds worth of data will be displayed, with one RMS value being calculated every 2 seconds. This results in 6 data points being displayed on the chart.

3.2.2.2.1.4 Combined View Screen

The Combined View screen exhibits the same features that have been outlined for the Raw Data screen, the FFT screen, and the RMS screen. The combined view simply combines these three plots into a single, convenient display. A screenshot of the Combined View screen can be seen in Figure 27 below.

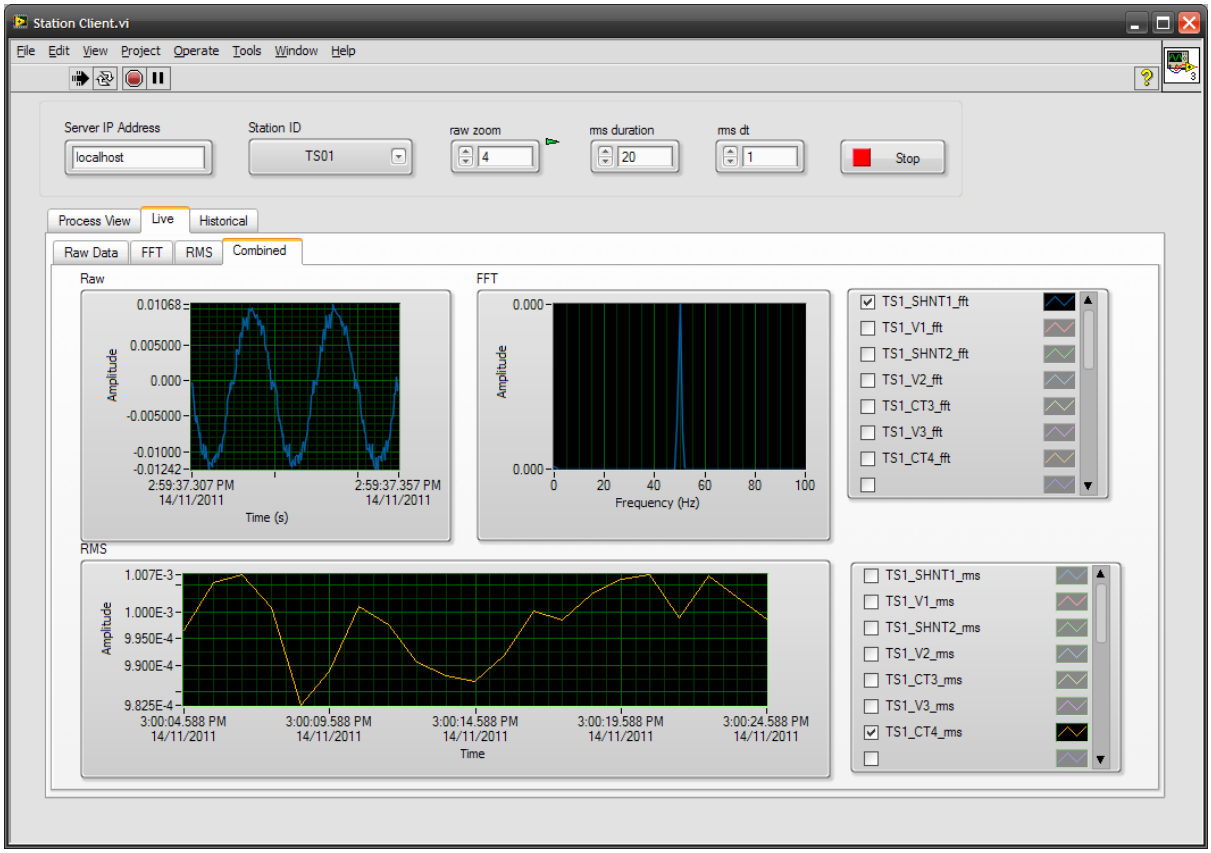

Figure 27 - Combined View Screen

3.2.2.2.2 Historical Data / Trending Screen

The Historical Data screen allows logged data to be requested from the server, and subsequently displayed within the client program. A screenshot of the Historical Data screen can be seen in Figure 28 below.

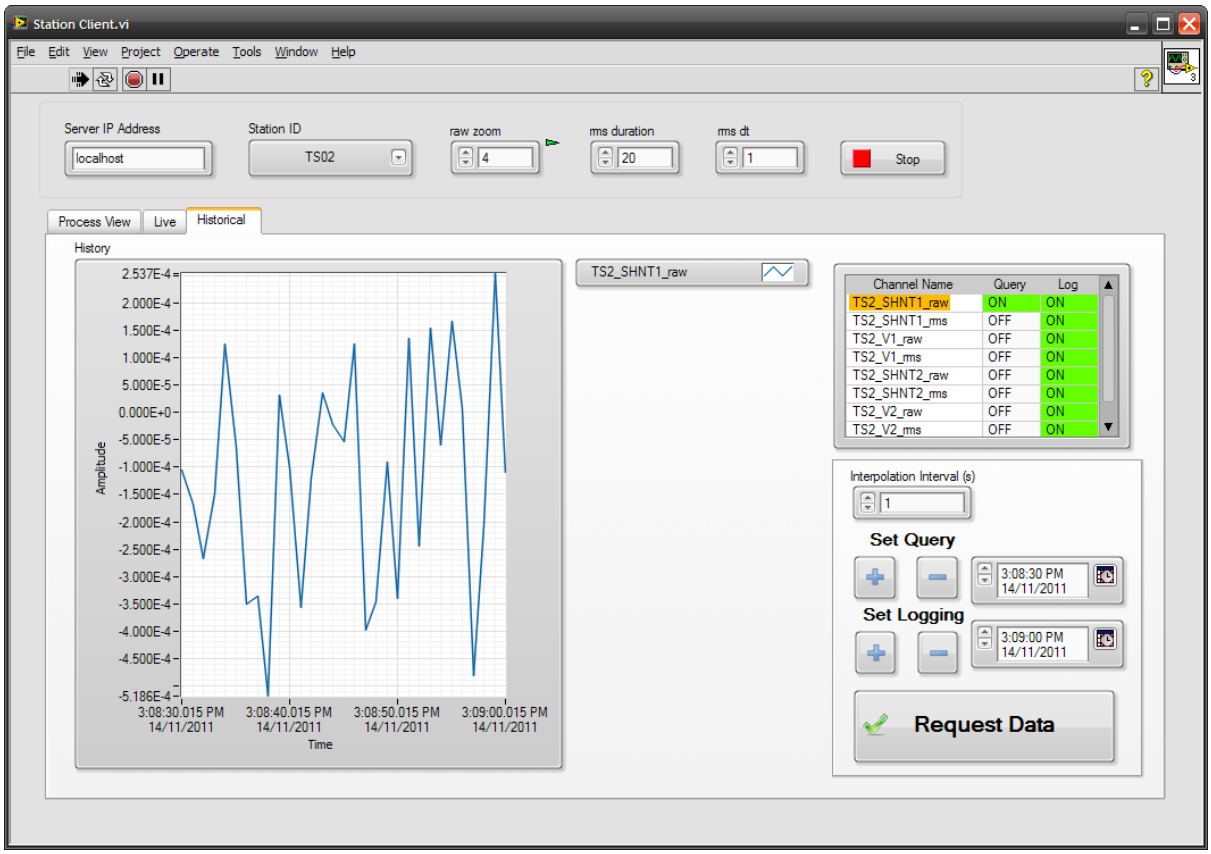

Figure 28 - Historical Data Screen

As can be seen from Figure 28 above, a variety of functionality is available from the Historical Data screen. A table is presented in the top right of the screen that allows logging and querying settings to be configured. Once logging has been turned on for a channel, the selected channel gets its value logged by the server. By default, logging is turned off for raw values, and turned on for RMS values. Channels that are being logged have their associated logging box coloured green, whereas channels that are not being logged are coloured white.

To request historical data from the server, the user must define the following settings:

- A start and end timestamp for the request.
- An interpolation interval for the data to be returned at.
- A selection of traces needing to be queried.

Start and end timestamps, as well as the interpolation interval are configurable from the box in the bottom right hand corner of the display. Channel queries are toggled on and off in the same way logging is turned on and off. Upon clicking the 'Request Data' button, a request is sent to the server to retrieve the data. Once the requested data is returned by the server, it is displayed within the client program.

3.3 Plans for Completion

At the time of this report's publication, a substantial amount of Curtin University's Green Electrical Energy Project has been completed. A meeting has recently been held with Curtin University staff, in which the following tasks were outlined for completion:

- The SMA Sunny Web Box is to be interfaced with LabVIEW so that data can be received from the SMA Windy Boy Inverter. (This requires the purchase of SMA's proprietary OPC server software.)
- Additional calculated channels (such as Power Factors) should be included within the software.
- The 'Process View' page should be completed on the client application.
- An 'Instructor's Application' should be developed. This application is to be similar to the current client application, but exhibit higher-level user privileges than the standard client software.
- Implement one or more remote client interfaces to allow a selected sub-set of system data to be accessed outside of Curtin University's local network.

Subsequent to an internal assessment of the tasks needing to be completed, it was determined that approximately 17 additional 'man days' worth of work are required to get the project to the commissioning stage.

4 Conclusion

Upon completion of the Murdoch University Internship Program, both the Pilot Plant Upgrade Project, and the Curtin Green Energy Project are in their software development stages. The Pilot Plant Upgrade project has currently been put on hold, but is in a state where work can resume following approval. The Curtin Green Energy Project is making steady progress, and is expected to be completed in the following few weeks. Throughout the internship program, numerous opportunities have been provided to apply the skills and professionalism taught at university to real-world engineering applications. The technical knowledge and industry skills that have been obtained through working with ICON Technologies will undoubtedly prove to be an invaluable asset throughout future work within the engineering industry.

5 Bibliography

Ritter, D. 2002. *LabVIEW® GUI : Essential Techniques*. Chicago: R.R. Donnelley and Sons Co. [1]

> *LabVIEW® GUI : Essential Techniques* explores a variety of factors relating to the graphical user interface (GUI) component of the LabVIEW platform. Guidance on how to create effective GUI's is presented, and many fundamental concepts (such as the psychology of human-computer interactions) are explained.

Johnson, G., Jennings, R. 2001. *LabVIEW® Graphical Programming*. Chicago: R.R. Donnelley and Sons Co. [2]

> *LabVIEW® Graphical Programming* provides a reference for many fundamental LabVIEW programming concepts, as well as a variety of advanced concepts such as data acquisition programs, and process control applications.

Essick, J. 1999. *Advanced LabVIEW® Labs*. New Jersey: Prentice-Hall, Inc. [3]

> *Advanced LabVIEW® Labs provides detailed explanations on how to perform various fundamental tasks in LabVIEW. Screenshots of relevant code is regularly presented and explained so that the reader is able to implement the expressed coding techniques in future applications.*

National Instruments. 2011. *http://www.ni.com/* (accessed 12/08/2011) [4]

> National Instrument's business website provides detailed information about all modern LabVIEW hardware and software available on the market. Consumer pricing is provided for each hardware component, and technical support links for National Instruments hardware/software is provided.

Bitter, R., Mohiuddin, T., Nawrocki, M. 2007. *LabVIEW® Advanced Programming Techniques (Second Edition)*. Washington: Taylor & Francis Group. [5]

> *LabVIEW® Advanced Programming Techniques (Second Edition)* provides a reference for many advanced LabVIEW programming techniques such as state machine implementation, and memory optimisation through code structure.

ICON Technologies. 2011. http://www.icon-tech.com.au/ (accessed 14/11/2011) [6]

> ICON Technologies' business website provides detailed information about ICON's role in the software industry, and how LabVIEW can be used to construct custom built software for clients.

Programmable Automation Controller. 2011. http://en.wikipedia.org/wiki/Programmable_automation_controller/ (accessed 13/11/2011) [7]

> Wikipedia contains concise, user-submitted content across a broad range of topics.

6 Appendices

6.1 Appendix 1 – Optimal Compact Rio Price Quote

With Breakout Boards				
Part Required		**Quantity Required**	**Spare I/O**	**Total Cost**
DI Module:	NI9425	4	24	$2,508.00
DO Module:	NI9476	3	28	$1,881.00
AI Module:	NI9208	3	13	$2,491.50
AO Module:	NI9265	6	2	$2,805.00
Chassis +	NI cRIO-9074	1	0	$3,635.50
Expansion Chassis:	NI9144	1	0	$1,347.50
			TOTAL COST:	**$14,668.50**

Table 3 - Compact Rio Price Quote (With Breakout Boards)

Without Breakout Boards				
Part Required		**Quantity Required**	**Spare I/O**	**Total Cost**
DI Module:	NI9421	13	0	$1,716.00
DO Module:	NI9472	9	4	$3,267.00
AI Module:	NI9203	5	5	$3,300.00
AO Module:	NI9265	6	2	$2,805.00
Chassis +	NI cRIO-9074	1	0	$3,635.50
Expansion Chassis:	NI9144	4	7	$5,390.00
			TOTAL COST:	**$20,113.50**

Table 4 - Compact Rio Price Quote (Without Breakout Boards)

Total Cost of System (With Breakout Boards):	**$14,668.50**
Total Cost of System (Without Breakout Boards):	**$20,113.50**

Table 5 - Compact Rio Price Quote Summary

Prices were retrieved from the National Instruments website:

National Instruments. 2011. http://www.ni.com/ (accessed 12/08/2011) [4]

6.2 Appendix 2 – Optimal Compact Fieldpoint Price Quote

Hardware FMG already owns:

	Model	Inputs (Per Unit)	Quantity Held	Total I/O	Cost (Per Unit)
Controller+BackPlane:	cFP-2220	8	1	8	$4,327.40
Expansion Chassis:	cFP-1808	8	1	8	$1,479.50
Digital Inputs:	cFP-DI-301	16	2	32	$308.00
Digital Outputs:	cFP-DO-401	16	4	64	$374.00
Analogue Inputs:	cFP-AI-110	8	3	24	$709.50
Analogue Outputs:	cFP-AO-210	8	1	8	$913.00
Connector Block:	cFP-CB-1		10		$225.50

Table 6 - FMG's Existing Hardware

Required additional components:

	Model	Inputs (Per Unit)	Additional Quantity Required	Total I/O (combined)	Cost (Per Unit)	Spare I/O Channels
Controller+BackPlane:	cFP-2220	8	0	8	$4,327.40	
Expansion Chassis:	cFP-1808	8	1	16	$1,479.50	4
Digital Inputs:	cFP-DI-301	16	5	112	$308.00	8
Digital Outputs:	cFP-DO-401	16	1	80	$374.00	12
Analogue Inputs:	cFP-AI-110	8	2	40	$709.50	5
Analogue Outputs:	cFP-AO-210	8	2	24	$913.00	2
Connector Block:	cFP-CB-1		10		$225.50	0

Total Cost of System:	$22,108.90
Total Additional Cost:	**$8,893.50**

Table 7 - Compact Fieldpoint Price Quote

Prices were retrieved from the National Instruments website:

National Instruments. 2011. http://www.ni.com/ (accessed 12/08/2011) [4]

6.3 Appendix 3 – Revised FMG Price Quote

Part	Part Number	Quantity	Cost (per unit) (+ GST)	Cost (+ GST)	Cost (Without GST)
Chassis + Controller	NI cRIO-9074	1	$3,635.50	$3,635.50	$3,305.00
Expansion Chasis	NI 9144	5	$1,347.50	$6,737.50	$6,125.00
Digital Input Module	NI 9421	13	$132.00	$1,716.00	$1,560.00
Digital Output Module	NI 9472	10	$363.00	$3,630.00	$3,300.00
Analogue Input Module	NI 9203	7	$660.00	$4,620.00	$4,200.00
Analogue Output Module	NI 9265	7	$467.50	$3,272.50	$2,975.00
Ethernet Switch	NI MES-3980	1	$1,479.50	$1,479.50	$1,345.00
Power Supply (24 VDC, 5A)	NI PS-15	7	$280.50	$1,963.50	$1,785.00
Filler Module	NI 9977	11	$38.50	$423.50	$385.00
			Total Cost:	**$27,478.00**	**$24,980.00**

Table 8 – Revised FMG Price Quote

6.4 Appendix 4 – HMI Screen Layouts

6.4.1 Screen 1:

REMOVED FOR COPYRIGHT PURPOSES

Figure 29 - FMG Pilot Plant Upgrade: HMI Screen 1

6.4.2 Screen 2:

REMOVED FOR COPYRIGHT PURPOSES

Figure 30 - FMG Pilot Plant Upgrade: HMI Screen 2

6.4.3 Screen 3:

REMOVED FOR COPYRIGHT PURPOSES

Figure 31 - FMG Pilot Plant Upgrade: HMI Screen 3

6.5 Appendix 5 – Green Electrical Energy Project – I/O List

Box	Chassis	Slot	Module	Channel	Description
1	1	1	9229	0	TS1-SHNT1
1	1	1	9229	1	TS1-V1
1	1	1	9229	2	TS1-SHNT2
1	1	1	9229	3	TS1-V2
1	1	2	9203	0	TS1-CT3
1	1	3	9229	0	TS1-V3
1	1	4	9229	0	TS2-SHNT1
1	1	4	9229	1	TS2-V1
1	1	4	9229	2	TS2-SHNT2
1	1	4	9229	3	TS2-V2
1	1	5	9203	0	IOB1-AS2
1	1	6	9977	N/A	N/A
1	1	7	9977	N/A	N/A
1	1	8	9977	N/A	N/A
1	2	1	9227	0	TS1-CT4
1	2	1	9227	1	TS1-CT5
1	2	2	9225	0	TS1-V4
1	2	2	9225	1	TS1-V5
1	2	3	9977	N/A	N/A
1	2	4	9977	N/A	N/A
1	2	5	9977	N/A	N/A
1	2	6	9977	N/A	N/A
1	2	7	9977	N/A	N/A
1	2	8	9977	N/A	N/A
2	1	1	9229	0	TS3-SHNT1
2	1	1	9229	1	TS3-V1
2	1	1	9229	2	TS3-SHNT2
2	1	1	9229	3	TS3-V2
2	1	2	9229	0	TS4-SHNT1
2	1	2	9229	1	TS4-V1
2	1	2	9229	2	TS4-SHNT2
2	1	2	9229	3	TS4-V2
2	1	3	9977	N/A	N/A
2	1	4	9977	N/A	N/A
2	1	5	9977	N/A	N/A
2	1	6	9977	N/A	N/A
2	1	7	9977	N/A	N/A
2	1	8	9977	N/A	N/A
2	2	1	9227	0	TS4-CT3
2	2	1	9227	1	TS4-CT4
2	2	2	9225	0	TS4-V3
2	2	2	9225	1	TS4-V4
2	2	3	9227	0	TS5-CT1
2	2	4	9225	0	TS5-V1
2	2	5	9977	N/A	N/A
2	2	6	9977	N/A	N/A
2	2	7	9977	N/A	N/A
2	2	8	9977	N/A	N/A

Box	Chassis	Slot	Module	Channel	Description
3	1	1	9227	0	6CT1-1
3	1	1	9227	1	6CT1-2
3	1	1	9227	2	6CT1-3
3	1	2	9225	0	6V1-1
3	1	2	9225	1	6V1-2
3	1	2	9225	2	6V1-3
3	1	3	9227	0	6CT2-1
3	1	3	9227	1	6CT2-2
3	1	3	9227	2	6CT2-3
3	1	4	9977	N/A	N/A
3	1	5	9227	0	CT3-1
3	1	5	9227	1	CT3-2
3	1	5	9227	2	CT3-3
3	1	6	9225	0	6V3-1
3	1	6	9225	1	6V3-2
3	1	6	9225	2	6V3-3
3	1	7	9227	0	6CT4-1
3	1	7	9227	1	6CT4-2
3	1	7	9227	2	6CT4-3
3	1	8	9225	0	6V4-1
3	1	8	9225	1	6V4-2
3	1	8	9225	2	6V4-3
4	1	1	9229	0	TS7-SHNT2
4	1	1	9229	1	TS7-V2
4	1	2	9871	P1	TS7-TS7-485
4	1	3	9977	N/A	N/A
4	1	4	9977	N/A	N/A
4	1	5	9977	N/A	N/A
4	1	6	9977	N/A	N/A
4	1	7	9977	N/A	N/A
4	1	8	9977	N/A	N/A
4	2	1	9227	0	TS7-CT1
4	2	1	9227	1	TS7-CT3
4	2	2	9225	0	TS7-V1
4	2	2	9225	1	TS7-V3
4	2	3	9977	N/A	N/A
4	2	4	9977	N/A	N/A
4	2	5	9977	N/A	N/A
4	2	6	9977	N/A	N/A
4	2	7	9977	N/A	N/A
4	2	8	9977	N/A	N/A

	Module Count					
Module:	**9229**	**9225**	**9227**	**9203**	**9977**	**9871**
Count:	6	7	8	2	32	1

Table 9 - Green Electrical Energy Project - I/O List